ARMY RESEARCH LABORATORY

A Method for Characterizing the Infrared Emissions From Kinetic Energy Penetrators

Thomas Kottke

ARL-MR-329

August 1996

NOTICES

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	August 1996	Final, Mar - Nov 95

4. TITLE AND SUBTITLE
A Method for Characterizing the Infrared Emissions From Kinetic Energy Penetrators

5. FUNDING NUMBERS
PR: 1L161102AH43

6. AUTHOR(S)
Thomas Kottke

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
U.S. Army Research Laboratory
ATTN: AMSRL-WT-WC
Aberdeen Proving Ground, MD 21005-5066

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-MR-329

9. SPONSORING/MONITORING AGENCY NAMES(S) AND ADDRESS(ES)

10.SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

A method is presented for characterizing the infrared (IR) emissions from kinetic energy (KE) penetrators. This two-step computer simulation method first generates a faceted surface model of the penetrator of interest and then computes the associated IR signature. The IR emission from each facet is individually computed and then the emissions from the facets are summed. This approach allows both the spectral distribution and spatial distribution of the IR radiation emission to be determined. The methods for generating a facet model and computing the radiometric quantities are presented in detail. Verification testing of this software is also demonstrated. These computer programs have been written to run on IBM-compatible personal computer (PC) platforms. In order to encourage the migration and application of these routines by other investigators, highly documented code listings of these modular programs have been included in the appendices.

14. SUBJECT TERMS
infrared (IR) emission, IR radiation, modeling, kinetic energy penetrator

15. NUMBER OF PAGES
85

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

INTENTIONALLY LEFT BLANK.

ACKNOWLEDGMENTS

The author would like to thank Mr. Charles R. Stumpfel of the Survivability Concepts Branch (SCB) for helpful discussions concerning radiometry theory and for carefully reviewing the technical details of this report along with Dr. Denis F. Strenzwilk (SCB). The author would also like to thank Ms. Mary Pfeiffer and Mrs. Tracey Mummert of LB&B Associates, Inc. for reviewing and polishing the final manuscript.

INTENTIONALLY LEFT BLANK.

TABLE OF CONTENTS

INTENTIONALLY LEFT BLANK.

LIST OF FIGURES

LIST OF TABLES

INTENTIONALLY LEFT BLANK.

1. INTRODUCTION

Temperatures of ballistic projectiles and kinetic energy (KE) penetrators in particular are elevated as they approach their impact points by in-bore heating during the launch phase and aerodynamic heating during the flight phase. The resulting emitted infrared (IR) radiation has been exploited as a means for tracking such penetrators (Thomson 1991). This method for projectile detection and tracking is particularly attractive because its inherently passive operation does not conflict with measures that may be employed to reduce armored vehicle signatures.

In order to utilize the IR radiation from penetrators for tracking purposes, it is necessary to characterize this radiation. Questions concerning the relative significance of various hot spots on the penetrator, the dependence of the IR signature on projectile orientation, the amount of signature variability introduced by projectile rotation, and requirements on detector and signal processing speed need to be addressed. This report presents a method for investigating these factors.

A two-step computer simulation process is presented that first generates a faceted surface model of the penetrator and then computes the associated IR signature by considering the IR emission from each individual facet. Both of these processes are presented in detail. A short primer is also included on radiometry. This should not be interpreted as an insult to the reader. Rather, this is presented as a common starting point in a field that is fraught with variability in both notation and definition.

In order to encourage the application of these routines by investigators in related studies, these programs have been written to run on IBM-compatible PC platforms. Highly documented code listings of these modular programs are included in the appendices.

2. PENETRATOR MODEL GENERATION

2.1 Penetrator Model Generation Overview. A first step in characterizing the IR energy emitted by a penetrator is to exercise an object generation program that creates a suitable computer model. Geometric details of the penetrator of interest are supplied as input to this program along with information about the temperature profile, thermal emissivities, orientation with

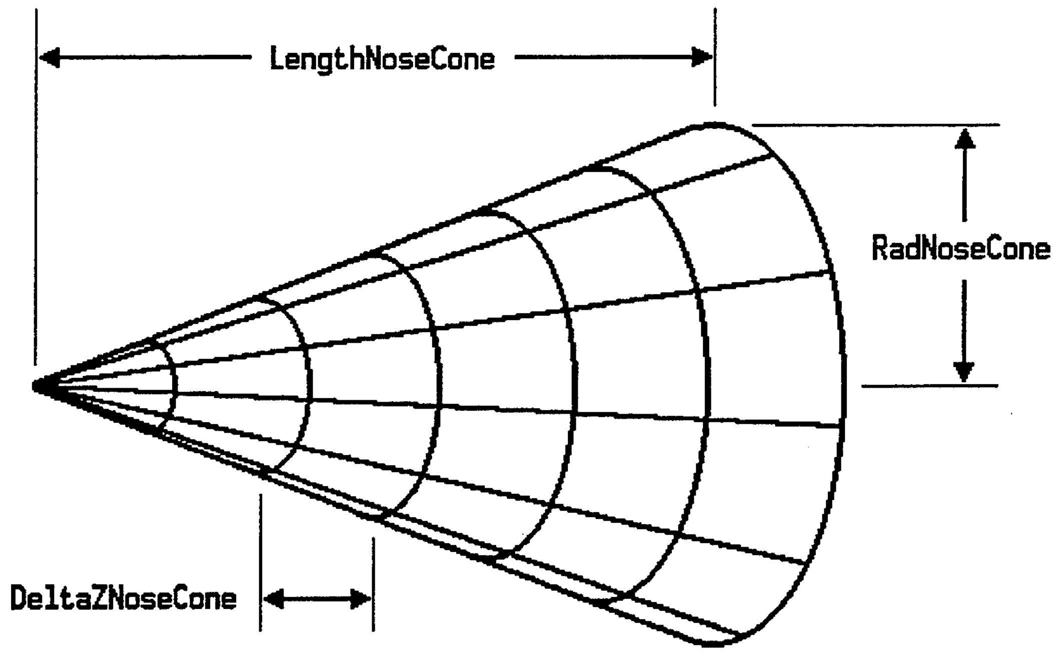
respect to the IR detector, and the manner in which the object is to be modeled. The program then generates a surface model of the penetrator that recreates the form as a collection of small planar surfaces, or facets. Each facet location and orientation is considered to determine whether or not it is visible at the location of the IR detector. A data file is constructed that records the location, orientation, area, temperature, and emissivity of all visible facets. This file can serve as the input to the IR spectrum calculation program that is described later.

The operation of the penetrator model generation software is now described in detail. A listing of this code is provided in Appendix A. Considerable effort has been expended to make the operation of this code readily understandable so that it may be easily modified and applied to a wide variety of applications. To achieve this goal, the program has been modularized through the use of subroutines, the listing has been extensively documented, and variable names have been utilized that describe their function or meaning. An unfortunate consequence of this explicit presentation is the considerable length of the resultant listing. The author hopes that the clarity of this code will offset any disadvantages resulting from its bulk.

Modeling of a penetrator-shaped object is simplified by the inherent simplicity and symmetry of the generic penetrator. In essence, most penetrators can be described as a conical forward section connected to a cylindrical body with equally spaced fins projecting radially at the rear. The object model generation software subdivides the penetrator into similar sections. Generation of the conical nose cone will now be described in detail. This will be followed by descriptions of how the cylindrical body, the fins, and the body areas between the fins are generated.

2.2 Conical Nose Cone Generation. Figure 1 illustrates the manner in which the conical nose cone section is modeled and the meaning of some of the relevant program variables. The overall dimensions of the conical section are defined by the length and base radius through the variables *LengthNoseCone* and *RadNoseCone*, respectively. Modeling resolution is determined by the variables *DeltaZNoseCone* and *NumNoseConeRadSeg%. DeltaZNoseCone* defines the thickness of each transverse nose cone "slice." *NumNoseConeRadSeg%* determines the number of radially oriented nose cone facets in each transverse nose cone slice.

Using these parameters, the subroutine *CalcNoseConePos* calculates the positions of each nose cone facet's corners, the area of each facet, and the components of each facet's outward



number of radial nose cone facet positions = NumNoseConeRadSeg%

Figure 1. Nose cone facet pattern and relevant program variables.

vector facing normal in body coordinates. Body coordinates are referenced to the penetrator with the z axis collocated with the penetrator's longitudinal axis and the x and y axes directed radially outward. The origin of the body coordinate system is taken as the tip of the nose cone with the positive z axis directed toward the rear of the penetrator. Calculation of the nose cone facet normal components is simplified by the fact that, due to the radial symmetry, the z component is the same for all of these facets.

The process of determining which nose cone facets are visible from the position of the IR detector is simplified by the assumption that the penetrator is always aimed toward the general direction of the IR detector. In this orientation, the conical nose cone will be closer to the detector than any other portion of the penetrator. Thus, no other portion of the penetrator can obscure any of the nose cone facets. Only the nose cone segments are capable of blocking other nose cone segments from the view of the IR detector. The simple symmetry of the conical nose cone allows the visibility of each facet to be determined by considering the orientation of the outward normal unit vector relative to the unit vector directed from the facet toward the IR detector. In particular, the dot product between these two unit vectors is computed. If this dot product is nonnegative,

then the facet's outward normal unit vector has a component directed toward the detector and the facet surface is visible to the detector. Facets for which this dot product is negative are facing away from the IR detector, and therefore the contribution of these facets to the overall IR spectrum can be disregarded.

2.3 Cylindrical Body Generation. The simple geometry of the cylindrical penetrator body allows the associated facet attributes to be readily calculated. This portion of the penetrator is assumed to extend from the base of the nose cone to the forwardmost fin position. Figure 2 displays the geometry of the cylindrical body and the significance of the relevant parameters. The length and radius of the penetrator body are defined by the parameters *LengthBody* and *RadBody*, respectively. Parameters *DeltaZBody* and *NumBodyRadSeg%* determine the size of each body facet and thus effectively control the resolution of the modeling for this portion of the penetrator.

The subroutine *CalcBodyPos* calculates the positions of the corners, the area, and the components of the outwardly facing normal unit vector for all the penetrator body facets in body coordinates. This task is simplified by the fact that the z component of the unit normal vectors is zero for all the body facets.

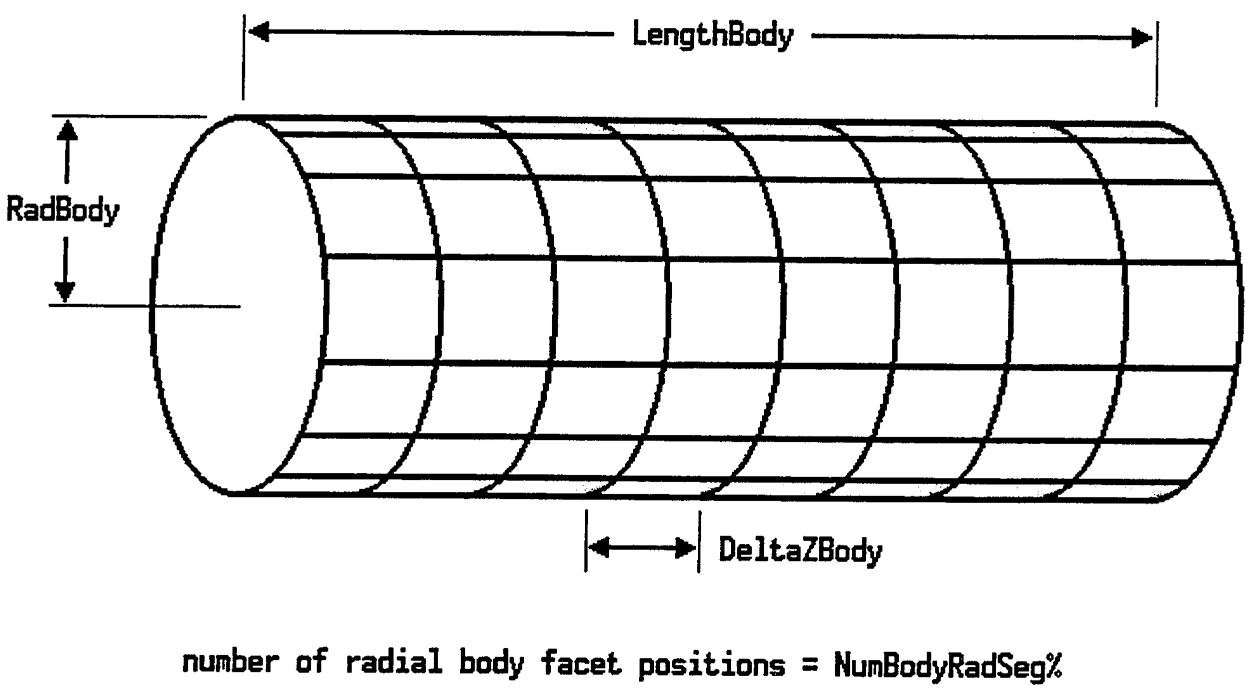


Figure 2. Body facet pattern and relevant program variables.

The assumption that the penetrator is aimed in the general direction of the IR detector also streamlines the process of determining which body facets are visible to the detector. Because the radius of any portion of the nose cone is always less than or equal to the body radius, the nose cone facets can never block any body facets that would otherwise be visible to the IR detector if the nose cone were removed. Therefore, the visibility of a body facet is also determined from the sign of the dot product between the facet's outward facing normal unit vector and the unit vector directed from the facet toward the detector.

2.4 <u>Fin Generation</u>. The fins are by far the most complex portion of the penetrator to be modeled. Figure 3 displays a generic fin and the parameters that define its shape and the manner in which a surface model is generated. The base length of the fin, where it attaches to the penetrator's body, is expressed by the variable *LengthBaseFin.* A forward portion of the fin is assumed to have a tapered profile. The extent of this tapered section is determined by the variable *LengthLeadEdgeFin. DeltaZFin* defines the size of the subdivisions along the z axis that the fin is partitioned into during the modeling process. The radial height of the fin is determined by the parameter *HeightFin,* and this length is subdivided into segments of dimension *DeltaRadFin.* Finally, the fin's thickness is quantified by the variable *ThickFin.* Facets along the edge of the fin that have a dimension defined by the variable *ThickFin* are referred to as edge facets. The remaining facets are referred to as side facets.

The corner positions, areas, and body coordinate components of the outward facing normal unit vectors are calculated by the subroutine *CalcFinPos* using the following simplifying observations.

a) All the edge facets in the tapered portion of the fin have the same normal unit vector z component.

b) All the edge facets in the tapered portion of the fin have the same area.

c) All the edge facets in the nontapered portion of the fin have a normal unit vector z component of zero.

d) All the edge facets in the nontapered portion of the fin have the same area.

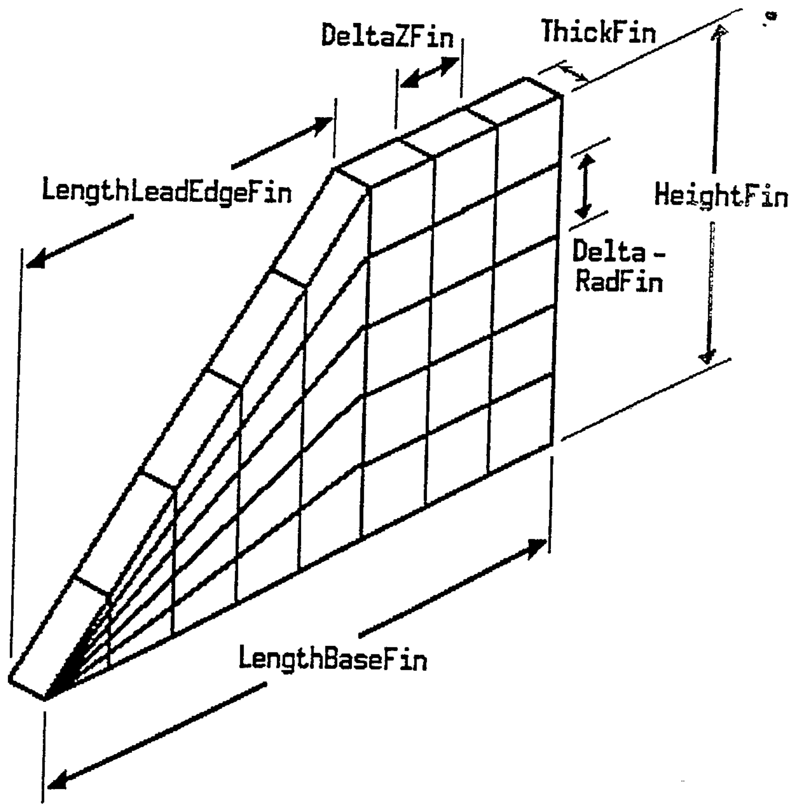


Figure 3. Fin facet pattern and relevant program variables.

e) Because the penetrator is assumed to be aimed in the general direction of the IR detector, the trailing edge of the fin will never be visible, and therefore the edge facets on the trailing edge need not be considered.

f) All the side facets have a normal unit vector z component of zero.

g) All the side facets on one side of a given fin have the same normal unit vector x and y components.

h) All the side facets in the nontapered portion of the fin have the same area.

The determination of which fin facets will be visible at the location of the IR detector is also somewhat involved. Recall that the visibility of the nose cone and body facets was determined by a simple vector dot product that, in essence, decided whether or not a facet surface pointed toward the IR detector. That is, the orientation of the detector relative to the penetrator precluded

the possibility that a facet in these portions of the penetrator could point toward the detector and still be blocked by another facet. This is not the case for the facets of the fin. Fin facets that point toward the detector may still not be visible to the detector because they are blocked by a facet in the nose cone, the body, the aftbody, or a facet on another fin. The aftbody portion of the penetrator has yet to be discussed. It is that portion of the penetrator body that is between the fins. Anyway, the subroutine *Eclipse* determines whether fin facets that point toward the detector are blocked by a facet in some other portion of the penetrator. Again, a number of observations are noted that streamline the associated calculations. Several of the following tests consider the fin's radial unit vector which is defined as the unit vector in the plane of the fin that is perpendicular to the z axis, or body, of the penetrator.

a) A nose cone, body, or aftbody facet will never block a fin facet for which the dot product of the fin's radial unit vector with the unit vector directed from the facet toward the detector is positive. Fin facets in this category must still be checked for blockage by facets on other fins.

b) A facet of a given fin with a surface that points toward the detector will never be blocked by another facet on that fin.

c) The facets on a fin can only be blocked by the facets on another fin for which the dot product between the fin radial unit vector and the unit vector directed from the facet toward the viewer has a value that is closer to 1. In other words, a fin facet can only be blocked by the facets of fins that are in front of it, as viewed by the detector.

d) An aftbody facet will never block a facet on a fin for which the dot product between the radial unit vector and the unit vector directed from the facet toward the viewer is nonnegative.

2.5 Aftbody Generation. The aftbody is that portion of the projectile body that is located between the fins. A representative aftbody area is illustrated in Figure 4. The overall longitudinal length and subdivision dimensions are the same as for the fins—namely *LengthBaseFin* and *DeltaZFin*, respectively. The tangential facet dimension is calculated from the circumference of the projectile body, the thickness of the fins, the total number of fins, and the number of radial facets between adjacent fins.

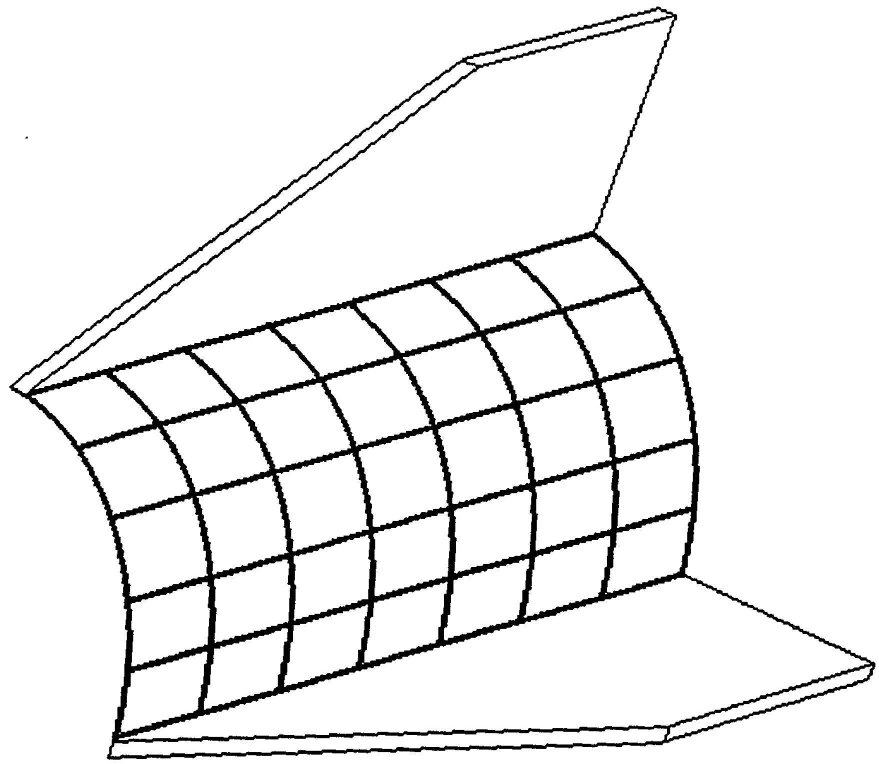


Figure 4. Aftbody facet pattern.

Aftbody facet corner positions, areas, and orientations, relative to the body coordinate system, are computed by the subroutine *CalcAftBodyPos*. These calculations are simplified by the fact that all the aftbody facets have the same area and a normal vector z component of zero. Each facet's orientation and the subroutine *Eclipse* are utilized to determine which aftbody facets are visible to the IR detector.

2.6 Penetrator Surface Model Graphic Display. After the facet corner positions and outward facing normal unit vector components have all been determined in the body coordinate system, an image of the penetrator is generated that illustrates the appearance of the penetrator from the vantage point of the IR detector. This process requires a transformation of all the facet corner positions from the body coordinate system of the penetrator to the space coordinate system of the detector. The origin of the space coordinate system is taken to be at the detector with the positive z axis extending horizontally to the right, the y axis extending vertically upward, and the x axis extending horizontally forward, relative to the detector's view of the penetrator. The transformation from body to space coordinates is accomplished by determining the Eulerian angles that relate the two coordinate systems and expressing the elements of the orthogonal transformation matrix as trigonometric functions of these angles (Goldstein 1950). Multiplication

of the body coordinate position vectors by the orthogonal transformation matrix yields the corresponding position vectors in the space coordinate system.

The perspective view of the penetrator is made more realistic by appropriate shading of the individual facets. Each facet position is displayed on the graphic image in a shade of gray that is determined from the facet's orientation relative to the viewpoint of the detector. In particular, the intensity of the shade of gray is proportional to the dot product between the facet's outward normal unit vector and the unit vector directed from the facet to the detector. This has the effect of imparting a dark coloration to facets that are seen on edge, while facets that are viewed face-on are displayed as a bright white. Thus, the illusion is created that light is glinting off appropriate portions of the penetrator from a light source that is collocated at the position of the detector. Although this effect is dramatic when displayed on a computer monitor, it does not show up well on printed hard copies. Therefore, a representative example of such an image is displayed in Figure 5 without the shading enhancement. This image was generated using the following penetrator input parameters.

nose cone length:	100 mm
nose cone longitudinal facet length:	10 mm
nose cone radius:	14 mm
number of nose cone radial facets:	24
body length:	420 mm
body longitudinal facet length:	20 mm
body radius:	14 mm
number of body radial facets:	24
number of fins:	6
fin thickness:	3 mm
fin total base length:	120 mm
fin leading edge base length:	80 mm
fin longitudinal facet length:	10 mm
fin height:	30 mm
fin transverse facet length:	5 mm
number of aftbody facets between adjacent fins:	4
Euler angles:	88.1°, 84.3°, 12°

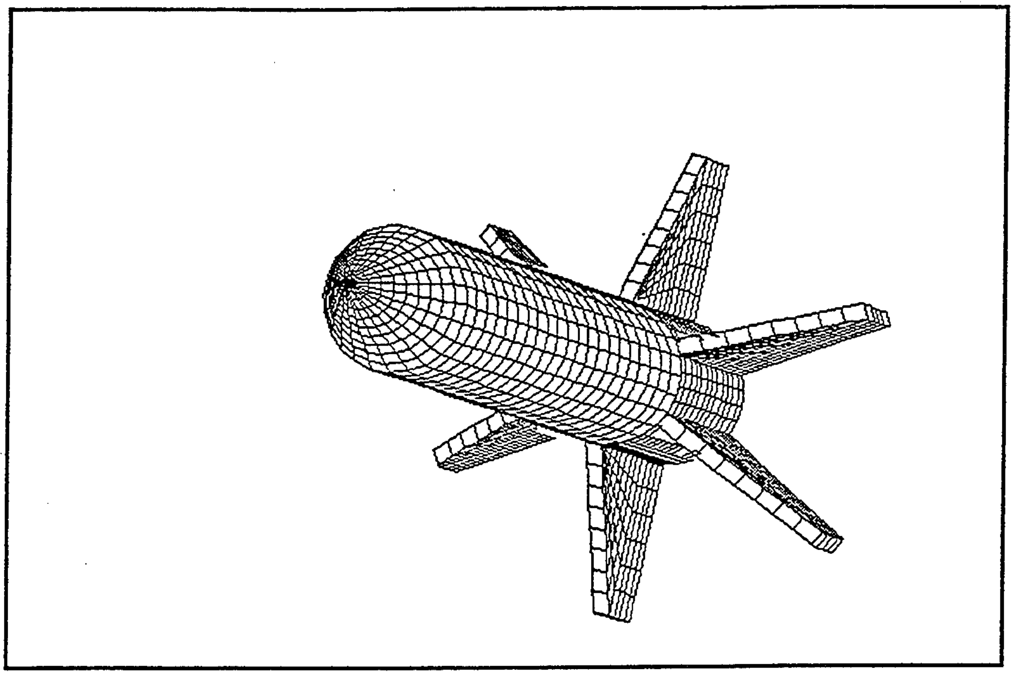


Figure 5. Example of penetrator surface model graphic display.

A graphical image of the penetrator is provided to the user as a quick "go/no go" check that the software is indeed modeling the intended scenario. Erroneous entry of penetrator dimension specifications are generally obvious from the resulting distorted image.

2.7 Penetrator Surface Model Temperature Assignment. The final computational task for the penetrator surface model generation software is the determination and allocation of a temperature to each facet. This is accomplished by interpolating between temperature values that are assigned to specific penetrator locations. In particular, the user must assign temperatures to the tip and base of the nose cone, the forward and rearmost body locations, the leading and trailing fin edges, the base fin position relative to the fin tip, and the forward and rearmost aftbody locations. Each facet's temperature is computed by determining its position relative to the appropriate defined temperature locations and interpolating a position-weighted intermediate temperature value. These calculations are handled by the subroutines *AssignNoseTemp*, *AssignBodyTemp*, *AssignFinSideTemp*, *AssignFinEdgeTemp*, and *AssignAftBodtTemp*. At the present time, these interpolations are a linear function of the facet's location between the defined boundary positions. In time, as computational or experimental determinations of actual penetrator flight temperatures become more quantitative, a more sophisticated interpolation process may be warranted. The modular nature of the temperature assignment subroutines will allow higher order interpolation techniques to be readily implemented.

A thermal emissivity is also assigned to each facet. This requires no computation. Rather, each facet within each region of the penetrator (nose cone, body, fin, or aftbody) is assumed to have the same direction-independent assigned thermal emissivity value. Directional emissivities are reserved for incorporation in future refinements of this process.

A data file is generated that contains the pertinent information about each penetrator surface facet. In particular, the recorded data include the region of the penetrator that the facet is located in, the facet corner positions in space coordinates, the value of the dot product between the facet's outward normal unit vector and the unit vector directed from the facet toward the detector, the area of the facet, the temperature of the facet, and the facet's thermal emissivity. Each data file includes an extensive leader that lists the geometric and thermal parameters that were used to generate the data set. This data file can be used as the input to the spectral calculation software that will be considered next.

3. PENETRATOR IR EMISSION CALCULATION

3.1 Penetrator IR Emission Calculation Overview. Recall that the goal of this modeling effort is to predict the amount and nature of the radiant energy that is generated by a heated penetrator and is collected by a suitable detector. To facilitate the required calculations, the penetrator has been modeled as a collection of surface facets. The approach is to separately consider the radiant energy that interacts with the detector from each facet and then sum these individual contributions to determine a total rate of radiant energy transfer. Effects arising from atmospheric absorption are not included due to the relatively short range over which IR sensors are expected to operate. The radiant energy transfer is characterized both spectrally and spatially. That is, the spectrum of the radiant energy that is emitted by various portions of the penetrator and incident upon the detector is determined. A detailed description of this process is now presented.

A newcomer to the field of radiometry is often bewildered by the abundance of terminology that, at first glance, appears to be describing the same thing. As an example, consider the fact that radiant energy, radiant energy density, radiant flux, radiant emittance, radiant photon emittance, radiant intensity, radiance, and irradiance are all terms that are commonly used to describe radiant energy transfer. In fact, this profusion of nomenclature arises from the need to consider a variety of scenarios involving radiation sources and detectors, both individually and

in combination, using a number of conventional normalization schemes. Luckily, a properly charted course will allow the reader, and even the author, to navigate this linguistic labyrinth.

The first task is to consider the spectral radiant emittance, which is the rate of radiant energy emission into a hemisphere per unit source area per unit wavelength interval at a particular wavelength. Already this is beginning to sound rather complicated. However, at this point the only considerations are the type and the rate at which radiant energy is coming off of a facet with no particular concern for where it is going. Clearly, the next step is to realize that the emitted radiation needs to go somewhere. This leads to a consideration of spectral radiance where not only the rate at which the radiant energy coming off a facet is addressed, but also the direction of that radiant energy. Up until this point, the facets have been treated as isolated sources. By finally introducing the detector into the scenario, the spectral irradiance can be calculated, which is a measure of the rate at which radiant energy from a facet is incident on the detector per unit detector area per unit wavelength interval at a particular wavelength. This is a general quantity because no assumptions have been made about the detector system. Additional radiometric quantities can be considered for specific detection apparatus.

If the collection area of the detector is known, the spectral radiant flux can be calculated. This is a measurement of the rate at which incident radiant energy enters the detector per unit wavelength interval at a particular wavelength. Of course, not all the radiant energy that enters the detector is necessarily recorded by the detector. If the spectral response of the detection system is known, then the total rate at which the detector records the incident radiant energy can be computed. This sounds a lot like the intermediate goal. So at this point a small victory will be declared, the same procedure will be followed for all the other facets, and all the individual facet contributions will be summed together. To keep things honest, the final result for a test case will be examined for validity to determine whether or not any celebrations were premature. So, with this radiometric roadmap in hand, the journey is begun by considering the spectral radiant emittance.

3.2 <u>Spectral Radiant Emittance</u>. All objects emit and absorb radiant energy. The quantity and character of this radiant energy depends on the temperature and nature of the object. A class of particularly convenient materials, known as black bodies, effectively absorb all incident radiant energy. Good absorbers of radiant energy also turn out to be good emitters. Thus, black

bodies are the best emitters of radiant energy. The spectral distribution of the radiant energy emitted by a black body is described by Planck's law,

$$W_\lambda = \frac{c_1}{\lambda^5} \frac{1}{e^{c_2/\lambda T} - 1} \tag{1}$$

where:

W_λ = spectral radiant emittance, $W \cdot cm^{-2} \cdot \mu m^{-1}$

λ = wavelength, μm

T = absolute temperature, K

C_1 = 3.742×10^4, $W \cdot cm^{-2} \cdot \mu m^4$

C_2 = 1.439×10^4, $\mu m \cdot K$.

In words, this expression calculates the radiant energy emitted by a black body of temperature T in a spectral band that is centered at wavelength λ per unit area per unit wavelength interval. Notice that this equation does not include any parameters that specify material properties of the emitting black body. This is in fact one of the conveniences associated with considering black body radiators. Black body radiation is totally dependent on the temperature of the black body and is totally independent of the particular material.

Unfortunately, most materials of unspecified configuration do not naturally exhibit black body characteristics. However, the spectral radiant emittance of objects that are not black bodies can be approximated by multiplying the expression of Equation 1 by an effective emissivity. An object's emissivity is a measure of its ability to emit radiant energy. Black bodies have an emissivity value of 1 while all other objects have an effective emissivity between 0 and 1. Emissivities can themselves be functions of both wavelength and temperature. Variations in spectral radiant emittance arising from an emissivity wavelength dependence are generally small compared to the strong wavelength dependence exhibited by Equation 1. The temperature dependence of the emissivity of representative metals has been shown to be weak over the temperature range 300–700 K (Snyder, Gier, and Dunkle 1955). It should also be noted that observed emissivity can depend on oxidation and mechanical surface treatments (Bramson 1968).

For all these reasons, and the fact that emissivity data in the literature is often sketchy at best, the use of a constant effective emissivity value is a common approximation.

3.3 Spectral Radiance. The next step is to consider where the radiant energy emitted by a facet is directed. At this point another common assumption is invoked that utilizes the angular distribution pattern of radiation emitted from perfectly diffuse sources. For such sources, known as Lambertian sources, the intensity of emitted radiation is proportional to the cosine of the angle between the surface normal and the emitted radiation direction of interest. When first encountered, this functional dependence for the radiation distribution pattern may appear somewhat arbitrary. However, recall that the effective projected area of a surface also depends on the cosine of the same angle. The spectral radiance of a perfectly diffuse source is therefore independent of viewing angle. A commonly cited example of a perfectly diffuse source is the sun. As predicted, the image of the sun appears to be uniformly bright in spite of the fact that the central portion is viewed face on while the edges are observed at a large angle from the solar surface normal.

Black bodies act as ideal diffuse sources. Real surfaces that are not black body radiators also tend to follow Lambert's cosine law quite closely at smaller viewing angles. However, at larger viewing angles, the amount of deviation from the ideal diffuse source radiation pattern is both material and surface topography dependent and can be significant (Hudson 1969).

As noted, one advantage in assuming a Lambertian source radiation pattern is the independence of the spectral radiance with respect to viewing angle. Another advantage is the fact that the spectral radiance from a surface radiating out into a hemisphere is related to the spectral radiant emittance by the simple relation

$$N_\lambda = W_\lambda / \pi \qquad (2)$$

where:

W_λ = spectral radiant emittance, $W \cdot cm^{-2} \cdot \mu m^{-1}$

N_λ = spectral radiance, $W \cdot cm^{-2} \cdot sr^{-1} \cdot \mu m^{-1}$.

At this point it is worth digressing for a quick cautionary note. As stated, the Lambertian surface is assumed to radiate energy into a complete hemisphere of space. A hemisphere contains 2π steradians of solid angle. Therefore, a common impulse is to assume that the spectral radiant emittance and spectral radiance are related by a factor of 2π rather than the prescribed factor of π. As noted by Hudson (1969), "Of all the mistakes a newcomer to radiometry may make, confusion over this factor of 2 is an odds-on favourite." Avoiding this pitfall, we press on.

3.4 Spectral Irradiance. Having quantified the rate and the distribution of energy emitted by our radiant source, we are in a position to introduce the detector into the scenario and consider the rate at which radiant energy of a particular wavelength is incident on the detector per unit detector area per unit wavelength interval. The spectral irradiance at the detector can be related to the spectral radiance of the source facet using the relation (Brown 1992)

$$H_\lambda = A_{source} \cdot \Omega_{det} \cdot N_\lambda / A_{det} \tag{3}$$

where:

H_λ = spectral irradiance, $W \cdot cm^{-2} \cdot \mu m^{-1}$

A_{source} = effective source area, cm^2

Ω_{det} = solid angle subtended by detector, sr

N_λ = spectral radiance, $W \cdot cm^{-2} \cdot sr^{-1} \cdot \mu m^{-1}$

A_{det} = effective detector area, cm^2.

The first multiplicative factor on the right-hand side of Equation 3 is the effective area of the source. In this case, the source is the penetrator model facet under consideration. and the effective area is the area of the facet as "seen" by the detector. This effective area, or projected area, is obtained by multiplying the actual area of the facet by the cosine of the angle between the facet's outward normal vector and the normal vector directed from the facet to the detector. In equation form this can be expressed as

$$A_{source} = (\hat{n} \cdot \hat{r}) \cdot A_f \qquad (4)$$

where:

A_{source} = effective source area, cm^2

$\hat{n}$ = facet normal unit vector

$\hat{r}$ = unit vector directed from facet to detector

A_f = facet area, cm^2.

Conveniently, the areas of the facets and their outward normal components are included in the data file of information that is created by the previously discussed penetrator model generation software.

The second multiplicative factor on the right-hand side of Equation 3 is the solid angle subtended by the detector. A common definition for solid angle is

$$d\Omega = \frac{1}{r^2} dA \qquad (5)$$

where:

$d\Omega$ = incremental solid angle

r = distance to the area

dA = incremental area.

From this expression it is easy to see why spheres, with surface areas of $4\pi r^2$, are associated with a solid angle of 4π steradians. Some of the confusion surrounding solid angles may result from the fact that although they are expressed in units of steradians, they are in fact dimensionless quantities—as can be concluded from a dimensional check of Equation 5. Anyway, in order to determine the solid angle subtended by the detector, the distance from the source facet to the detector and the effective area of the detector must be known. For purposes

of calculation, the effective detector area is assumed to be 1 cm^2. Therefore, the spectral irradiance can be expressed as

$$H_\lambda = (\hat{n} \cdot \hat{r}) \cdot A_f \cdot \frac{1}{r^2} \cdot N_\lambda \tag{6}$$

where:

H_λ	=	spectral irradiance, $W \cdot cm^{-2} \cdot \mu m^{-1}$
$\hat{n}$	=	facet normal unit vector
$\hat{r}$	=	unit vector directed from facet to detector
A_f	=	facet area, cm^2
r	=	distance from facet to detector, cm
N_λ	=	spectral radiance, $W \cdot cm^{-2} \cdot \mu m^{-1}$.

This formulation is general in the sense that it characterizes the nature of the radiation that is incident on the detector without making any assumptions about the detector itself.

3.5 Spectral Radiant Flux. Equation 6 is a generalized expression for the spectral irradiance at the site of a detector that is a distance r from a source facet. This is a measure of the rate at which energy is transferred to a unit area by radiation that is incident on that surface and that spans a specific interval of wavelengths. For a specific detector of known effective area, the spectral radiant flux can be determined. This is a measure of the rate at which radiant energy is conveyed to the detector position per unit wavelength interval at a particular wavelength. The spectral radiant flux can be determined from the spectral irradiance using the expression

$$P_\lambda = A_{det} \cdot H_\lambda \tag{7}$$

where:

P_λ = spectral radiant flux, $W \cdot \mu m^{-1}$

A_{det} = detector area, cm^2

H_λ = spectral irradiance, $W \cdot cm^{-2} \cdot \mu m^{-1}$.

For application to specific systems where the detection spectral response is known, the total rate at which the detector accepts radiant energy, or total radiant flux, can be determined. A method for determining this quantity is now presented.

3.6 Total Radiant Flux. Having arrived at an expression for the rate at which radiant energy is incident on the detector at a particular wavelength, it is now possible to consider the total rate at which the detection system acquires radiant energy. This requires a knowledge of the spectral response of the detector system. In this case, the detector system response is assumed to include the effects of all focusing and filtering elements in addition to the conversion efficiency of the detector itself. The total radiant flux is computed by summing the spectral radiant flux, modulated by the detection system response, over a band of wavelengths that corresponds to the detector's active region.

Recall that the spectral radiant flux is defined as the rate of transfer of radiant energy per unit wavelength interval at a particular wavelength. Within a particular wavelength interval the spectral radiant flux is assumed to be constant. The strong wavelength dependence exhibited by Equation 1 suggests that this assumption is only valid over a narrow range of wavelengths. Therefore, in practice, the radiant emittance, radiance, irradiance, and radiant flux of broad wavelength regions are calculated by subdividing the region into many narrow subregions, calculating the spectral quantity of each subregion, and then summing the individual results. For a wavelength region extending from λ_1 to λ_2 that is divided into N subregions with central wavelength values of λ_i and wavelength spans of $(\Delta\lambda)_i$, the total radiant flux for a detector system with spectral response η_{λ_i} can be expressed as

$$P(\lambda_1 \rightarrow \lambda_2) = \sum_{i=1}^{N} \eta_{\lambda_i} \cdot P_{\lambda_i} \cdot (\Delta\lambda)_i . \tag{8}$$

At this point, one of the stated goals has been achieved. A method has been outlined for calculating the rate of radiant energy transfer, or radiant flux, between a single facet and a detector with a specified spectral response. The next step is to calculate the same quantity for various sections of the penetrator. Because the penetrator has been modeled as a collection of facets, this step is straightforward. The radiant flux is separately calculated for all the facets in a region of interest, and the individual results are summed together to yield the total radiant flux. The spectral analysis software that performs these functions is listed in Appendix B.

3.7 IR Emission Analysis Software Verification. The spectral analysis software is verified by comparing its results against the output from a commercially available software package that can calculate the IR emission spectrum for simple geometries. Integrated Sensors (Integrated Sensors, Inc. 1989) offers a disk-based IR spectrum calculator that computes the total flux and black body spectrum for sources with a specified temperature, range of emission wavelengths, emissivity, and field of view. Careful selection of input parameters for the penetrator generation and spectral analysis software presented in this report and Integrated Sensor's IR spectrum calculator can yield equivalent scenarios for which the outputs can be directly compared.

One such scenario is a circular source with a 10-mm radius located 10 m from a 1-cm^2 detector that accepts IR energy in the wavelength band from 2.0 μm to 5.5 μm. The following penetrator generation and spectrum analysis input parameters were used to create this case.

nose cone length	10 mm
nose cone longitudinal facet length	1 mm
nose cone radius	10 mm
number of nose cone radial facets	50
body length	20 mm
body longitudinal facet length	10 mm
body radius	10 mm
number of body radial facets	50
number of fins	1
fin thickness	1 mm
fin base length	10 mm
fin leading edge length	6 mm
fin longitudinal facet length	2 mm
fin height	4 mm

fin radial facet length	2 mm
number of aftbody facets between fins	1
Euler angles	90, 90, –90
nose cone tip temperature	573 K, 873 K, or 1,173 K
nose cone rear temperature	same as nose cone tip temperature
nose cone emissivity	0.1
body forward temperature	0 K
body rear temperature	0 K
body emissivity	0
fin leading edge temperature	1 K
fin trailing edge temperature	0 K
fin emissivity	0
aftbody forward temperature	0 K
aftbody rear temperature	0 K
aftbody emissivity	0
IR emission band	2.0 μm to 5.5 μm
calculation spectral window width	0.01 μm
penetrator range	10 m

These parameters yield a 10-mm radius penetrator that is aimed directly toward the detector. Thus, the nose cone appears circular to the detector. The extraneous body, fin, and aftbody penetrator components are effectively eliminated from the IR emission calculation by their assignment of very low temperatures and emissivities of 0. Three test cases are considered with nose cone temperatures of 573 K, 863 K, and 1,173 K.

The following inputs to Integrated Sensor's black body calculator yielded the equivalent scenario.

temperature	573 K, 873 K, or 1,173 K
wavelength band starting value	2.0 μm
wavelength band ending value	5.5 μm
emissivity	0.1
detector solid angle	1 E –6
source area	3.1416 cm^2

The results of the radiant flux calculations for the common scenario as determined using these two different programs are presented in Table 1.

Table 1. Comparison of Verification Radiant Flux Calculations

Temperature (K)	Results From Software Presented in This Report	Results From Integrated Sensor's Black Body Calculator
573	1.87E-8 W	1.87E-8 W
873	1.89E-7 W	1.89E-7 W
1,173	6.92E-7 W	6.91E-7 W

The close correlation between the results of these two IR emission calculation programs indicates that the results of the software presented in this report are numerically valid.

4. SUMMARY

A method is presented for characterizing the IR emissions from KE penetrators. Descriptions of this type are required for the application of IR tracker systems where questions concerning apparent source location, orientation effects, rotation effects, and detection speed need to be addressed. This two-step computer simulation method first generates a faceted surface model of the penetrator of interest and then computes the associated IR signature. The IR emission from each facet is individually computed. This approach allows both the spectral distribution and spatial distribution of the IR radiation emission to be determined. The methods for generating a facet model and computing the radiometric quantities are presented in detail. Verification testing of this software is also demonstrated.

These computer programs have been written to run on IBM-compatible PC platforms. In order to encourage the migration and application of these routines by other investigators, highly documented code listings of these modular programs have been included in the appendices. Future reports will highlight the results of IR characterization studies of specific scenarios.

INTENTIONALLY LEFT BLANK.

5. REFERENCES

Bramson, M. A. Infrared Radiation, a Handbook for Applications. New York: Plenum Press, 1968.

Brown, T. G. Radiometry and Detection. New York: University of Rochester, 1992.

Goldstein, H. Classical Mechanics. Reading, MA: Addison-Wesley, 1950.

Hudson, Jr., R. D. Infrared System Engineering. New York: Wiley-Interscience, 1969.

Integrated Sensors, Inc. "Black Body Calculator, Version 1.3." Goleta, CA, 1989.

Snyder, N. W., J. T. Gier, and R. V. Dunkle. "Total Normal Emissivity Measurements on Aircraft Materials Between 100 and 800° F." Transactions of the ASME, pp. 1011–1019, October 1955.

Thomson, G. M. "Passive IR Tracking of Incoming Kinetic Energy Munitions." Proceedings of the 1991 BRL Technical Symposium: Emerging Technology for the Future Battlefield, Aberdeen Proving Ground, MD, November 1991.

INTENTIONALLY LEFT BLANK.

APPENDIX A:

PENETRATOR MODEL GENERATION SOFTWARE

INTENTIONALLY LEFT BLANK.

This software generates a facet model of a kinetic energy (KE) penetrator that can subsequently be used to calculate the spatial and spectral distributions of associated infrared (IR) emissions. MicroSoft QuickBasic 4.5 is used as the programming environment. If you have any questions about this code, please contact Tom Kottke at:

AMSRL-WT-WD
Survivability Concepts Branch
Weapons Concepts Division, Bldg. 120
Weapons Technology Directorate
U.S. Army Research Laboratory
Aberdeen Proving Ground, MD 21005-5066
(410) 278-2557

```
                                     'the size of the arrays is allowed to
                                     'change as required during execution
REM $DYNAMIC
                                     'subroutines are declared
DECLARE SUB InputParameters ()
DECLARE SUB CalcNoseConePos ()
DECLARE SUB CalcBodyPos ()
DECLARE SUB CalcFinPos ()
DECLARE SUB CalcBoxExtremes ()
DECLARE SUB CalcEulerElem ()
DECLARE SUB TransCoordBS (XOld, YOld, ZOld, XSpace, YSpace, ZSpace)
DECLARE SUB TransCoordSB (XSpace, YSpace, ZSpace, XBody, YBody, ZBody)
DECLARE SUB CalcScreenSize ()
DECLARE SUB Init3DDisplay ()
DECLARE SUB DisplayBox ()
DECLARE SUB Plot3DPoint (X, Y, Z, C1)
DECLARE SUB Plot3DLine (X, Y, Z, C1)
DECLARE SUB Eclipse (XTest, YTest, ZTest, FinNumber%, FinUnitCOSValue, Type$, Ans%)
DECLARE SUB CalcFinEdgeMidPoint (FinNumber%, LeadEdgeLongSegNumber%, XMid, YMid,
        ZMid)
DECLARE SUB CalcFinSideMidPoint (FinNumber%, LeadEdgeLongSegNumber%,
        FinRadSegNumber%, SideNumber%, XMid, YMid, ZMid)
DECLARE SUB CalcAftBodyPos ()
DECLARE SUB CalcAftBodyMidPoint (ZSegment%, FinNumber%, AftBodyRadSegNumber%, XMid,
        YMid, ZMid)
DECLARE SUB LoadDataFile (Type$, X(), Y(), Z(), NormalDotProduct, Area, Temp, Emis)
DECLARE SUB AssignNoseTemp (ZSegment%, NoseTemp)
DECLARE SUB AssignBodyTemp (ZSegment%, BodyTemp)
DECLARE SUB AssignFinSideTemp (LeadEdgeLongSegNumber%, FinRadSegNumber%, FinTemp)
DECLARE SUB AssignFinEdgeTemp (LeadEdgeLongSegNumber%, FinTemp)
DECLARE SUB AssignAftBodyTemp (ZSegment%, AftBodyTemp)

C1 = 1
DIM Euler(3)
                                     'projectile parameters are input
CALL InputParameters
                                     'array variables are dimensioned
DIM ZNoseConePos(NumNoseConeLongSeg%, 4)
DIM XNoseConePos(NumNoseConeLongSeg%, NumNoseConeRadSeg%, 4)
DIM YNoseConePos(NumNoseConeLongSeg%, NumNoseConeRadSeg%, 4)
DIM NoseConeArea(NumNoseConeLongSeg%)
DIM XNoseConeNormal(NumNoseConeRadSeg%)
DIM YNoseConeNormal(NumNoseConeRadSeg%)
DIM ZBodyPos(NumBodyLongSeg%, 4)
```

```
DIM XBodyPos(NumBodyRadSeg%, 4)
DIM YBodyPos(NumBodyRadSeg%, 4)
DIM XBodyNormal(NumBodyRadSeg%)
DIM YBodyNormal(NumBodyRadSeg%)
DIM E(3, 3)
DIM ZFinEdgePos(NumLeadEdgeLongSeg% + NumNonLeadEdgeLongSeg%, 4)
DIM YFinEdgePos(NumFins%, NumLeadEdgeLongSeg% + NumNonLeadEdgeLongSeg%, 4)
DIM XFinEdgePos(NumFins%, NumLeadEdgeLongSeg% + NumNonLeadEdgeLongSeg%, 4)
DIM ZFinPos(NumLeadEdgeLongSeg% + NumNonLeadEdgeLongSeg%, 4)
DIM YFinPos(NumFins%, NumLeadEdgeLongSeg% + NumNonLeadEdgeLongSeg%,
                                                NumFinRadSeg%, 2, 4)
DIM XFinPos(NumFins%, NumLeadEdgeLongSeg% + NumNonLeadEdgeLongSeg%,
                                                NumFinRadSeg%, 2, 4)
DIM XFinLeadEdgeNormal(NumFins%), XFinNonLeadEdgeNormal(NumFins%)
DIM YFinLeadEdgeNormal(NumFins%), YFinNonLeadEdgeNormal(NumFins%)
DIM XFinSideNormal(NumFins%, 2), YFinSideNormal(NumFins%, 2)
DIM ZAftBodyPos(NumLeadEdgeLongSeg% + NumNonLeadEdgeLongSeg%, 4)
DIM XAftBodyPos(NumFins%, NumAftBodyRadSegPerFin%, 4)
DIM YAftBodyPos(NumFins%, NumAftBodyRadSegPerFin%, 4)
DIM XAftBodyNormal(NumFins%, NumAftBodyRadSegPerFin%)
DIM YAftBodyNormal(NumFins%, NumAftBodyRadSegPerFin%)
DIM X(4), Y(4), Z(4), FinLeadSideArea(NumLeadEdgeLongSeg%)

ScreenWidth = 638
ScreenHeight = 398
AspectRatio = .97
                                                'nose cone facet coordinates, areas, and
                                                'orientations are calculated
CALL CalcNoseConePos
                                                'body facet coordinates, areas, and
                                                'orientations are calculated
CALL CalcBodyPos
                                                'fin facet coordinates, areas, and
                                                'orientations are calculated
CALL CalcFinPos
                                                'afterbody facet coordinates, areas, and
                                                'orientations are calculated
CALL CalcAftBodyPos
                                                'determining the extreme positions
CALL CalcBoxExtremes
                                                'the matrix elements that are necessary
                                                'to make the transformations between
                                                'projectile and space coordinates are
                                                'calculated
CALL CalcEulerElem
                                                'the components of the normal vector
                                                'pointing from the projectile's body
                                                'coordinate origin to the viewer are
                                                'calculated in the body coordinate
                                                'system. note that the viewer's (or the
                                                'computer monitor's) coordinate system
                                                'has the positive x axis pointing into
                                                'the screen, the positive y axis pointing
                                                'up, and the positive z axis pointing
                                                'toward the right.
CALL TransCoordSB(-1, 0, 0, XViewNormal, YViewNormal, ZViewNormal)
                                                'an appropriate scale for the graphic
                                                'display screen is determined
CALL CalcScreenSize
                                                'the graphic display is initialized, the
                                                'graphic display parameters are stored in
                                                'the data file, and the display colors
                                                'are defined
CALL Init3DDisplay
```

```
'*****************************************************************************************
'THIS NEXT SECTION OF CODE PLOTS OUT ALL THE PROJECTILE FACETS USING THE
'SUBTLE CONSTRUCTION LINE COLOR. LATER, THE VISIBLE FACETS WILL BE REDRAWN
'USING A COLOR THAT REPRESENTS THEIR ORIENTATION RELATIVE TO THE VIEWER.
'*****************************************************************************************
                                                    'all the nose cone facets are plotted

                                                    'each transverse nose cone slice is
                                                    'considered
FOR ZSegment% = 0 TO NumNoseConeLongSeg% - 1
                                                    'each radial facet within the transverse
                                                    'slice is considered
  FOR CordSegment% = 0 TO NumNoseConeRadSeg% - 1
                                                    'the first corner of each facet is plotted
                                                    'as a point
    CALL TransCoordBS(XNoseConePos(ZSegment%, CordSegment%, 4),
                      YNoseConePos(ZSegment%, CordSegment%, 4),
                      ZNoseConePos(ZSegment%, 4), XSpace, YSpace, ZSpace)
    CALL Plot3DPoint(XSpace, YSpace, ZSpace, 2)
                                                    'lines are drawn connecting all the
                                                    'corners
    FOR Corner% = 1 TO 4
      CALL TransCoordBS(XNoseConePos(ZSegment%, CordSegment%, Corner%),
                        YNoseConePos(ZSegment%, CordSegment%, Corner%),
                        ZNoseConePos(ZSegment%, Corner%), XSpace, YSpace,
                        ZSpace)
      CALL Plot3DLine(XSpace, YSpace, ZSpace, 2)
      NEXT Corner%

    NEXT CordSegment%
  NEXT ZSegment%
                                                    'all the body facets are plotted

                                                    'each transverse body slice is considered
FOR ZSegment% = 0 TO NumBodyLongSeg% - 1
                                                    'each radial facet within the transverse
                                                    'slice is considered
  FOR CordSegment% = 0 TO NumBodyRadSeg% - 1
                                                    'the first corner of each facet is plotted
                                                    'as a point
      CALL TransCoordBS(XBodyPos(CordSegment%, 4), YBodyPos(CordSegment%, 4),
                        ZBodyPos(ZSegment%, 4), XSpace,
                        YSpace, ZSpace)
      CALL Plot3DPoint(XSpace, YSpace, ZSpace, 2)
                                                    'lines are drawn connecting all the
                                                    'corners
      FOR Corner% = 1 TO 4
        CALL TransCoordBS(XBodyPos(CordSegment%, Corner%),
                          YBodyPos(CordSegment%, Corner%),
                          ZBodyPos(ZSegment%, Corner%), XSpace,
                          YSpace, ZSpace)
        CALL Plot3DLine(XSpace, YSpace, ZSpace, 2)
        NEXT Corner%

    NEXT CordSegment%
  NEXT ZSegment%
                                                    'all the fin facets are plotted

                                                    'each fin is considered in turn
FOR FinNumber% = 0 TO NumFins% - 1
                                                    'each transverse fin slice is considered
  FOR LeadEdgeLongSegNumber% = 0 TO NumLeadEdgeLongSeg% +
                          NumNonLeadEdgeLongSeg% - 1

                                                    'the side facets in each transverse
```

```
                                        'slice are considered
    FOR FinRadSegNumber% = 0 TO NumFinRadSeg% - 1
                                        'both of the fin sides are considered
      FOR SideNumber% = 1 TO 2
                                        'the first facet corner is plotted as
                                        'a point
        CALL TransCoordBS(XFinPos(FinNumber%, LeadEdgeLongSegNumber%,
                          FinRadSegNumber%, SideNumber%, 4),
                          YFinPos(FinNumber%, LeadEdgeLongSegNumber%,
                          FinRadSegNumber%, SideNumber%, 4),
                          ZFinPos(LeadEdgeLongSegNumber%, 4), XSpace, YSpace,
                          ZSpace)
        CALL Plot3DPoint(XSpace, YSpace, ZSpace, 2)
                                        'lines are drawn connecting all the
                                        'corners
        FOR Corner% = 1 TO 4
          CALL TransCoordBS(XFinPos(FinNumber%, LeadEdgeLongSegNumber%,
                          FinRadSegNumber%, SideNumber%, Corner%),
                          YFinPos(FinNumber%, LeadEdgeLongSegNumber%,
                          FinRadSegNumber%, SideNumber%, Corner%),
                          ZFinPos(LeadEdgeLongSegNumber%,
                          Corner%), XSpace, YSpace, ZSpace)
          CALL Plot3DLine(XSpace, YSpace, ZSpace, 2)
          NEXT Corner%

        NEXT SideNumber%
      NEXT FinRadSegNumber%
                                        'the leading edge facet is considered

                                        'the first facet corner is plotted as
                                        'a point
    CALL TransCoordBS(XFinEdgePos(FinNumber%, LeadEdgeLongSegNumber%, 4),
                          YFinEdgePos(FinNumber%, LeadEdgeLongSegNumber%, 4),
                          ZFinEdgePos(LeadEdgeLongSegNumber%, 4), XSpace,
                          YSpace, ZSpace)
    CALL Plot3DPoint(XSpace, YSpace, ZSpace, 2)
                                        'lines are drawn connecting all the
                                        'corners
    FOR Corner% = 1 TO 4
      CALL TransCoordBS(XFinEdgePos(FinNumber%, LeadEdgeLongSegNumber%,
                          Corner%), YFinEdgePos(FinNumber%,
                          LeadEdgeLongSegNumber%, Corner%),
                          ZFinEdgePos(LeadEdgeLongSegNumber%, Corner%), XSpace,
                          YSpace, ZSpace)
      CALL Plot3DLine(XSpace, YSpace, ZSpace, 2)
      NEXT Corner%

    NEXT LeadEdgeLongSegNumber%
  NEXT FinNumber%
                                        'all the aftbody facets are plotted

                                        'each transverse aftbody slice is
                                        'considered
FOR ZSegment% = 0 TO NumLeadEdgeLongSeg% + NumNonLeadEdgeLongSeg% - 1
                                        'the afterbody region between each pair
                                        'of adjacent fins is considered
  FOR FinNumber% = 0 TO NumFins% - 1
                                        'each afterbody radial facet is considered
    FOR AftBodyRadSegNumber% = 0 TO NumAftBodyRadSegPerFin% - 1
                                        'the first facet corner is plotted as
                                        'a point
      CALL TransCoordBS(XAftBodyPos(FinNumber%, AftBodyRadSegNumber%, 4),
                          YAftBodyPos(FinNumber%, AftBodyRadSegNumber%, 4),
                          ZAftBodyPos(ZSegment%, 4), XSpace, YSpace, ZSpace)
      CALL Plot3DPoint(XSpace, YSpace, ZSpace, 2)
                                        'lines are drawn connecting all the
```

```
                                                'corners
          FOR Corner% = 1 TO 4
             CALL TransCoordBS(XAftBodyPos(FinNumber%, AftBodyRadSegNumber%,
                                Corner%), YAftBodyPos(FinNumber%,
                                AftBodyRadSegNumber%, Corner%),
                                ZAftBodyPos(ZSegment%, Corner%),
                                XSpace, YSpace, ZSpace)
             CALL Plot3DLine(XSpace, YSpace, ZSpace, 2)
             NEXT Corner%

          NEXT AftBodyRadSegNumber%
        NEXT FinNumber%
      NEXT ZSegment%

'****************************************************************************************************
'EACH PROJECTILE FACET IS RECONSIDERED TO DETERMINE WHETHER IT IS IN A
'POSITION THAT IS VISIBLE TO THE VIEWER. THOSE FACETS THAT ARE VISIBLE ARE
'REDRAWN USING A COLOR THAT REPRESENTS THE FACETS ORIENTATION WITH RESPECT
'TO THE VIEWER. FACETS WITH A NORMAL VECTOR THAT POINTS DIRECTLY TOWARD THE
'VIEWER ARE REPLOTTED USING WHITE LINES. FACETS WITH A NORMAL VECTOR THAT IS
'PERPENDICULAR TO THE VIEWERS DIRECTION ARE REPLOTTED USING BLACK LINES.
'FACETS WITH NORMAL VECTORS THAT FALL BETWEEN THESE TWO EXTREMES ARE
'REPLOTTED USING AN APPROPRIATE SHADE OF GRAY. THIS COLORING SCHEME YIELDS AN
'IMAGE OF THE PROJECTILE THAT MIMICS THE CASE WHERE THE ILLUMINATING LIGHT
'SOURCE IS BETWEEN THE VIEWER'S EYES.
'****************************************************************************************************

                                                'all the nose cone facets are reconsidered

                                                'each transverse nose cone slice is
                                                'considered
    FOR ZSegment% = 0 TO NumNoseConeLongSeg% - 1
                                                'each radial facet within the transverse
                                                'slice is considered
      FOR CordSegment% = 0 TO NumNoseConeRadSeg% - 1
                                                'the visibility of a facet to the viewer
                                                'is determined by considering the dot
                                                'product between the facet's outwardly
                                                'pointed normal vector and the normal
                                                'vector pointing towards the viewer. if
                                                'this dot product is positive, then the
                                                'facet is visible. it is assumed that no
                                                'other portion of the projectile will
                                                'ever block a nose cone facet.
        NormalDotProduct = XNoseConeNormal(CordSegment%) * XViewNormal +
                                YNoseConeNormal(CordSegment%) * YViewNormal +
                                ZNoseConeNormal * ZViewNormal
        IF (NormalDotProduct > 0) THEN
                                                'a corner of each visible facet is
                                                'plotted as a point
          CALL TransCoordBS(XNoseConePos(ZSegment%, CordSegment%, 4),
                                YNoseConePos(ZSegment%, CordSegment%, 4),
                                ZNoseConePos(ZSegment%, 4), XSpace, YSpace, ZSpace)
          CALL Plot3DPoint(XSpace, YSpace, ZSpace, NormalDotProduct * 12 + 3)
                                                'lines are drawn connecting the facer
                                                'corners
          FOR Corner% = 1 TO 4
             CALL TransCoordBS(XNoseConePos(ZSegment%, CordSegment%, Corner%),
                                YNoseConePos(ZSegment%, CordSegment%, Corner%),
                                ZNoseConePos(ZSegment%, Corner%), XSpace, YSpace,
                                ZSpace)
             CALL Plot3DLine(XSpace, YSpace, ZSpace, NormalDotProduct * 12 + 3)
                                                'facet corner positions are saved for
                                                'later transfer to the data storage file
             X(Corner%) = XSpace
             Y(Corner%) = YSpace
```

```
          Z(Corner%) = ZSpace
          NEXT Corner%
                                              'the temperature of the facet is
                                              'determined by interpolation
        CALL AssignNoseTemp(ZSegment%, NoseTemp)
                                              'data is transferred to the storage file
        CALL LoadDataFile("Nose", X(), Y(), Z(), NormalDotProduct,
                          NoseConeArea(ZSegment%), NoseTemp, NoseEmis)
        END IF
      NEXT CordSegment%
    NEXT ZSegment%
                                              'all the body facets are reconsidered

                                              'each trasverse body slice is considered
  FOR ZSegment% = 0 TO NumBodyLongSeg% - 1
                                              'each radial facet within the transverse
                                              'slice is considered
    FOR CordSegment% = 0 TO NumBodyRadSeg% - 1
                                              'the visibility of a facet to the viewer
                                              'is determined by considering the dot
                                              'product between the facets outwardly
                                              'pointed normal vector and the unit
                                              'vector pointing towards the viewer. if
                                              'this dot product is positive, then the
                                              'facet is visible. it is assumed that no
                                              'other portion of the projectile will
                                              'ever block a body facet.
      NormalDotProduct = XBodyNormal(CordSegment%) * XViewNormal +
                          YBodyNormal(CordSegment%) * YViewNormal +
                          ZBodyNormal * ZViewNormal
      IF (NormalDotProduct > 0) THEN
                                              'the first corner of each visible facet
                                              'is plotted as a point
        CALL TransCoordBS(XBodyPos(CordSegment%, 4), YBodyPos(CordSegment%, 4),
                          ZBodyPos(ZSegment%, 4), XSpace, YSpace, ZSpace)
        CALL Plot3DPoint(XSpace, YSpace, ZSpace, NormalDotProduct * 12 + 3)
                                              'lines are drawn connecting all the
                                              'facet corners
        FOR Corner% = 1 TO 4
          CALL TransCoordBS(XBodyPos(CordSegment%, Corner%),
                          YBodyPos(CordSegment%, Corner%), ZBodyPos(ZSegment%,
                          Corner%), XSpace, YSpace, ZSpace)
          CALL Plot3DLine(XSpace, YSpace, ZSpace, NormalDotProduct * 12 + 3)
                                              'facet corner positions are saved for
                                              'later transfer to the storage data file
          X(Corner%) = XSpace
          Y(Corner%) = YSpace
          Z(Corner%) = ZSpace
          NEXT Corner%
                                              'the temperature of the facet is
                                              'determined by interpolation
        CALL AssignBodyTemp(ZSegment%, BodyTemp)
                                              'data is transferred to the storage file
        CALL LoadDataFile("Body", X(), Y(), Z(), NormalDotProduct, BodyArea, BodyTemp,
                          BodyEmis)
        END IF
      NEXT CordSegment%
    NEXT ZSegment%
                                              'all the fin facets are reconsidered

                                              'each fin is considered in turn
  FOR FinNumber% = 0 TO NumFins% - 1
                                              'the x and y components in the body
                                              'coordinate system are calculated for
                                              'a unit vector lying in the plane of the
                                              'fin that is normal to the longitudinal
```

```
                                                     'axis
FinUnitRadXVector = -COS(2 * 3.14159 * FinNumber% / NumFins%)
FinUnitRadYVector = SIN(2 * 3.14159 * FinNumber% / NumFins%)
                                                     'this unit vector is then dotted with the
                                                     'unit vector pointing towards the viewer.
                                                     'the resulting cosine value is a measure
                                                     'of the degree to which the fin points
                                                     'toward the viewer
FinUnitCOSValue = FinUnitRadXVector * XViewNormal + FinUnitRadYVector *
                        YViewNormal
                                                     'each transverse fin slice is considered
FOR LeadEdgeLongSegNumber% = 0 TO NumLeadEdgeLongSeg% +
                        NumNonLeadEdgeLongSeg% - 1
                                                     'the side facets in each transverse
                                                     'slice are considered
   FOR FinRadSegNumber% = 0 TO NumFinRadSeg% - 1
                                                     'both of the fin sides are considered
      FOR SideNumber% = 1 TO 2
                                                     'the visibility of a facet to the viewer
                                                     'is determined by first considering the
                                                     'dot product between the facet's outward
                                                     'pointed normal vector and the normal
                                                     'vector pointed towards the viewer. if
                                                     'this dot product is positive then the
                                                     'facet may be visible to the viewer.
         NormalDotProduct = XFinSideNormal(FinNumber%, SideNumber%) * XViewNormal     +
YFinSideNormal(FinNumber%, SideNumber%) *                            YViewNormal +
ZFinSideNormal * ZViewNormal
         IF (NormalDotProduct > 0) THEN
                                                     'the areal midpoint position of the facet
                                                     'is calculated in the body coordinate
                                                     'system
            CALL CalcFinSideMidPoint(FinNumber%, LeadEdgeLongSegNumber%,
                        FinRadSegNumber%, SideNumber%, XMid, YMid, ZMid)
                                                     'this midpoint position is then
                                                     'transformed to the space coordinate
                                                     'system
            CALL TransCoordBS(XMid, YMid, ZMid, XSpace, YSpace, ZSpace)
                                                     'the graphic display plot color at this
                                                     'midpoint pixel position is saved
            OldColor = POINT(ZSpace * Scale, YSpace * Scale)
                                                     'the midpoint pixel position is replotted
                                                     'in white to denote the facet under
                                                     'consideration
            CALL Plot3DPoint(XSpace, YSpace, ZSpace, 15)
                                                     'all other projectile facets are
                                                     'considered to determine whether any of
                                                     'them will block the viewer's view of
                                                     'this facet
            CALL Eclipse(XSpace, YSpace, ZSpace, FinNumber%, FinUnitCOSValue, "Fin",
                        Ans%)
                                                     'if the facet is not blocked by any other
                                                     'facet then the subroutine returns with
                                                     'the variable Ans% equal to 1
            IF (Ans% = 1) THEN
                                                     'the midpoint position pixel is replotted
                                                     'in its original graphic display color
               CALL Plot3DPoint(XSpace, YSpace, ZSpace, OldColor)
                                                     'a corner of the facet is plotted as a
                                                     'point
               CALL TransCoordBS(XFinPos(FinNumber%, LeadEdgeLongSegNumber%,
                        FinRadSegNumber%, SideNumber%, 4),
                        YFinPos(FinNumber%, LeadEdgeLongSegNumber%,
                        FinRadSegNumber%, SideNumber%, 4),
                        ZFinPos(LeadEdgeLongSegNumber%, 4), XSpace, YSpace,
                        ZSpace)
```

```
            CALL Plot3DPoint(XSpace, YSpace, ZSpace, NormalDotProduct * 12 + 3)
                                               'lines are drawn between the facet corners
                                               'using a display color that indicates the
                                               'facets orientation with respect to the
                                               'viewer
            FOR Corner% = 1 TO 4
              CALL TransCoordBS(XFinPos(FinNumber%, LeadEdgeLongSegNumber%,
                         FinRadSegNumber%, SideNumber%, Corner%),
                         YFinPos(FinNumber%, LeadEdgeLongSegNumber%,
                         FinRadSegNumber%, SideNumber%, Corner%), ZFinPos(
                         LeadEdgeLongSegNumber%, Corner%), XSpace, YSpace,
                         ZSpace)

              CALL Plot3DLine(XSpace, YSpace, ZSpace, NormalDotProduct * 12 + 3)
                                               'the facet corner positions are saved for
                                               'later transfer to the data storage file
              X(Corner%) = XSpace
              Y(Corner%) = YSpace
              Z(Corner%) = ZSpace
              NEXT Corner%
                                               'the temperature of the facet is
                                               'determined by interpolation and the
                                               'data is transferred to the storage file
            IF (LeadEdgeLongSegNumber% < NumLeadEdgeLongSeg%) THEN
              CALL AssignFinSideTemp(LeadEdgeLongSegNumber%, FinRadSegNumber%,
                         FinTemp)
              CALL LoadDataFile("Fin ", X(), Y(), Z(), NormalDotProduct,
                         FinLeadSideArea(LeadEdgeLongSegNumber%), FinTemp,
                         FinEmis)
              ELSE
              CALL AssignFinSideTemp(LeadEdgeLongSegNumber%, FinRadSegNumber%,
                         FinTemp)
              CALL LoadDataFile("Fin ", X(), Y(), Z(), NormalDotProduct,
                         FinNonLeadSideArea, FinTemp, FinEmis)
              END IF
                                               'if the view of the facet is blocked by
                                               'another facet the midpoint position
                                               'pixel is simple returned to its
                                               'original color
            ELSE
            CALL Plot3DPoint(XSpace, YSpace, ZSpace, OldColor)
            END IF
          END IF
        NEXT SideNumber%
      NEXT FinRadSegNumber%
                                               'the fin edge for this longitudinal fin
                                               'slice is considered

                                               'the dot product of the edge facets
                                               'outward normal vector with the unit
                                               'vector towards the viewer is calculated
    IF (LeadEdgeLongSegNumber% < NumLeadEdgeLongSeg%) THEN
      NormalDotProduct = XFinLeadEdgeNormal(FinNumber%) * XViewNormal +
                         YFinLeadEdgeNormal(FinNumber%) * YViewNormal +
                         ZFinLeadEdgeNormal * ZViewNormal
      ELSE
      NormalDotProduct = XFinNonLeadEdgeNormal(FinNumber%) * XViewNormal +
                         YFinNonLeadEdgeNormal(FinNumber%) * YViewNormal
      END IF
                                               'this edge facet can only be visible if
                                               'the dot product is positive
    IF (NormalDotProduct > 0) THEN
                                               'the midpoint of the edge facet is
                                               'calculated in body coordinates
      CALL CalcFinEdgeMidPoint(FinNumber%, LeadEdgeLongSegNumber%, XMid, YMid,
                         ZMid)
```

```
                                        'this midpoint position is transformed to
                                        'space coordinates
CALL TransCoordBS(XMid, YMid, ZMid, XSpace, YSpace, ZSpace)
                                        'the original color of this midpoint
                                        'pixel position is saved
OldColor = POINT(ZSpace * Scale, YSpace * Scale)
                                        'the midpoint position pixel is replotted
                                        'in white to denote which facet is under
                                        'consideration
CALL Plot3DPoint(XSpace, YSpace, ZSpace, 15)
                                        'all other projectile facets are
                                        'considered to determine whether any of
                                        'them will block the viewer's view of
                                        'this facet
CALL Eclipse(XSpace, YSpace, ZSpace, FinNumber%, FinUnitCOSValue, "Fin", Ans%)
                                        'if the facet is not blocked by any other
                                        'facet then the subroutine returns with
                                        'the variable Ans% equal to 1
IF (Ans% = 1) THEN
                                        'the midpoint position pixel is replotted
                                        'in the original color
   CALL Plot3DPoint(XSpace, YSpace, ZSpace, OldColor)
                                        'a facet corner position is plotted as
                                        'a point
   CALL TransCoordBS(XFinEdgePos(FinNumber%, LeadEdgeLongSegNumber%, 4),
                     YFinEdgePos(FinNumber%, LeadEdgeLongSegNumber%, 4),
                     ZFinEdgePos(LeadEdgeLongSegNumber%, 4), XSpace,
                     YSpace, ZSpace)
   CALL Plot3DPoint(XSpace, YSpace, ZSpace, NormalDotProduct * 12 + 3)
                                        'lines are drawn between the facet corners
                                        'using a display color that indicates the
                                        'facets orientation with respect to the
                                        'viewer
   FOR Corner% = 1 TO 4
      CALL TransCoordBS(XFinEdgePos(FinNumber%, LeadEdgeLongSegNumber%,
                     Corner%), YFinEdgePos(FinNumber%,
                     LeadEdgeLongSegNumber%, Corner%),
                     ZFinEdgePos(LeadEdgeLongSegNumber%, Corner%), XSpace,
                     YSpace, ZSpace)
      CALL Plot3DLine(XSpace, YSpace, ZSpace, NormalDotProduct * 12 + 3)
                                        'the facet corner positions are saved for
                                        'later transfer to the data storage file
      X(Corner%) = XSpace
      Y(Corner%) = YSpace
      Z(Corner%) = ZSpace
      NEXT Corner%
                                        'the temperature of the facet is
                                        'determined by interpolation and the
                                        'data is transferred to the storage file
   IF (LeadEdgeLongSegNumber% < NumLeadEdgeLongSeg%) THEN
      CALL AssignFinEdgeTemp(LeadEdgeLongSegNumber%, FinTemp)
      CALL LoadDataFile("Fin ", X(), Y(), Z(), NormalDotProduct, FinLeadEdgeArea,
                     FinTemp, FinEmis)
      ELSE
      CALL AssignFinEdgeTemp(LeadEdgeLongSegNumber%, FinTemp)
      CALL LoadDataFile("Fin ", X(), Y(), Z(), NormalDotProduct, FinNonLeadEdgeArea,
                     FinTemp, FinEmis)
      END IF
                                        'if the view of the facet is blocked by
                                        'another facet the midpoint position
                                        'pixel is simple returned to its
                                        'original color
   ELSE
   CALL Plot3DPoint(XSpace, YSpace, ZSpace, OldColor)
   END IF
END IF
```

```
      NEXT LeadEdgeLongSegNumber%
   NEXT FinNumber%
                                                'all the aftbody facets are reconsidered

                                                'each transverse aftbody slice is
                                                'considered
FOR ZSegment% = 0 TO NumLeadEdgeLongSeg% + NumNonLeadEdgeLongSeg% - 1
                                                'the aftbody area between each pair of
                                                'adjacent fins is considered
   FOR FinNumber% = 0 TO NumFins% - 1
                                                'the radial facets within each transverse
                                                'aftbody slice are considered
      FOR AftBodyRadSegNumber% = 0 TO NumAftBodyRadSegPerFin% - 1
                                                'the visibility of a facet to the viewer
                                                'is determined by first considering the
                                                'dot product between the facets outwardly
                                                'pointed normal vector and the unit
                                                'vector pointing towards the viewer. if
                                                'this dot product is positive, then the
                                                'facet may be visible provided that it is
                                                'not blocked by another facet.
         NormalDotProduct = XAftBodyNormal(FinNumber%, AftBodyRadSegNumber%) *
                        XViewNormal + YAftBodyNormal(FinNumber%,
                        AftBodyRadSegNumber%) * YViewNormal
         IF (NormalDotProduct > 0) THEN
                                                'the areal midpoint position of the
                                                'aftbody facet is calculated in the
                                                'body coordinate system
            CALL CalcAftBodyMidPoint(ZSegment%, FinNumber%, AftBodyRadSegNumber%,
                        XMid, YMid, ZMid)
                                                'this midpoint position is tranformed to
                                                'the space coordinate system
            CALL TransCoordBS(XMid, YMid, ZMid, XSpace, YSpace, ZSpace)
                                                'the original color of the midpoint
                                                'position pixel is determined and saved
            OldColor = POINT(ZSpace * Scale, YSpace * Scale)
                                                'the aftbody midpoint position is replotted
                                                'in white to denote which facet is under
                                                'consideration
            CALL Plot3DPoint(XSpace, YSpace, ZSpace, 15)
                                                'all other projectile facets are
                                                'considered to determine whether any of
                                                'them will block the viewer's view of
                                                'this facet
            CALL Eclipse(XSpace, YSpace, ZSpace, FinNumber%, FinUnitCOSValue, "AftBody",
                        Ans%)
                                                'if the facet is not blocked by any other
                                                'facet then the subroutine returns with
                                                'the variable Ans% equal to 1
            IF (Ans% = 1) THEN
                                                'the aftbody facet midpoint position
                                                'pixel is returned to its original color
               CALL Plot3DPoint(XSpace, YSpace, ZSpace, OldColor)
                                                'a facet corner position is plotted as
                                                'a point
               CALL TransCoordBS(XAftBodyPos(FinNumber%, AftBodyRadSegNumber%, 4),
                        YAftBodyPos(FinNumber%, AftBodyRadSegNumber%, 4),
                        ZAftBodyPos(ZSegment%, 4), XSpace, YSpace, ZSpace)
               CALL Plot3DPoint(XSpace, YSpace, ZSpace, NormalDotProduct * 12 + 3)
                                                'lines are drawn between the facet corners
                                                'using a display color that indicates the
                                                'facets orientation with respect to the
                                                'viewer
               FOR Corner% = 1 TO 4
                  CALL TransCoordBS(XAftBodyPos(FinNumber%, AftBodyRadSegNumber%,
                        Corner%), YAftBodyPos(FinNumber%,
```

```
                                AftBodyRadSegNumber%, Corner%),
                                ZAftBodyPos(ZSegment%, Corner%), XSpace, YSpace,
                                ZSpace)
                CALL Plot3DLine(XSpace, YSpace, ZSpace, NormalDotProduct * 12 + 3)
                                           'the facet corner positions are saved for
                                           'later transfer to the data storage file
                X(Corner%) = XSpace
                Y(Corner%) = YSpace
                Z(Corner%) = ZSpace
                NEXT Corner%
                                           'the temperature of the aftbody facet is
                                           'determined by interpolation
              CALL AssignAftBodyTemp(ZSegment%, AftBodyTemp)
                                           'data is transferred to the storage file
              CALL LoadDataFile("Aft ", X(), Y(), Z(), NormalDotProduct, AftBodyArea,
                                AftBodyTemp, AftBodyEmis)
                                           'if the aftbody facet is blocked by
                                           'another facet then the midpoint position
                                           'pixel is simply returned to its original
                                           'color
              ELSE
              CALL Plot3DPoint(XSpace, YSpace, ZSpace, OldColor)
              END IF
            END IF
          NEXT AftBodyRadSegNumber%
        NEXT FinNumber%
      NEXT ZSegment%
                                           'the data storage file is terminated
                                           'and closed
   PRINT #1, "END "
   CLOSE #1
                                           'a completion statement is output to the
                                           'monitor screen
   LOCATE 27, 2
   PRINT "DONE"
                                           'a user terminated do loop is executed to
                                           'allow the graphic image to emain on the
                                           'monitor screen for possible graphic
                                           'capture
DO
   LOOP WHILE INKEY$ = ""

REM $STATIC
SUB AssignAftBodyTemp (ZSegment%, AftBodyTemp)
'*****************************************************************************************

'THIS SUBROUTINE DETERMINES AFTBODY TEMPERATURES BY LINEARLY INTERPOLATING
'BETWEEN THE PREVIOUSLY DEFINED TEMPERATURES AT THE FRONT AND REAR OF THE
'AFTBODY SECTION

'*****************************************************************************************

   SHARED NumLeadEdgeLongSeg%, NumNonLeadEdgeLongSeg%
   SHARED AftBodyForwTemp, AftBodyRearTemp

   AftBodyTemp = AftBodyForwTemp + (AftBodyRearTemp - AftBodyForwTemp) * (ZSegment%        /
(NumLeadEdgeLongSeg% + NumNonLeadEdgeLongSeg% - 1))

END SUB

SUB AssignBodyTemp (ZSegment%, BodyTemp)
```

```
'*************************************************************************************************************

'THIS SUBROUTINE DETERMINES BODY TEMPERATURES BY LINEARLY INTERPOLATING
'BETWEEN THE PREVIOUSLY DEFINED TEMPERATURES AT THE FRONT AND REAR OF THE
'BODY SECTION

'*************************************************************************************************************

   SHARED NumBodyLongSeg%, BodyForwTemp, BodyRearTemp

   BodyTemp = BodyRearTemp + (BodyForwTemp - BodyRearTemp) * ((NumBodyLongSeg% - 1)     -
ZSegment%) / (NumBodyLongSeg% - 1)

END SUB

SUB AssignFinEdgeTemp (LeadEdgeLongSegNumber%, FinTemp)

'*************************************************************************************************************

'THIS SUBROUTINE DETERMINES FIN EDGE TEMPERATURES BY LINEARLY INTERPOLATING
'BETWEEN THE PREVIOUSLY DEFINED TEMPERATURES AT THE INNERMOST AND
OUTERMOST
'FIN EDGE POSITIONS

'*************************************************************************************************************

   SHARED NumLeadEdgeLongSeg%, NumNonLeadEdgeLongSeg%, FinOuterWRTInnerTemp
   SHARED FinLeadTemp, FinTrailTemp

   IF (LeadEdgeLongSegNumber% < NumLeadEdgeLongSeg%) THEN
      FinTemp = FinLeadTemp + ((LeadEdgeLongSegNumber%) / (NumLeadEdgeLongSeg% - 1)) *
                                        FinOuterWRTInnerTemp
      ELSE
      FinTemp = FinLeadTemp + FinOuterWRTInnerTemp
      END IF

END SUB

SUB AssignFinSideTemp (LeadEdgeLongSegNumber%, FinRadSegNumber%, FinTemp)

'*************************************************************************************************************

'THIS SUBROUTINE DETERMINES FIN SIDE TEMPERATURES BY LINEARLY INTERPOLATING
'BETWEEN THE PREVIOUSLY DEFINED TEMPERATURES AT THE INNERMOST, OUTERMOST,
'LEADING AND TRAILING EDGE POSITIONS

'*************************************************************************************************************

   SHARED NumLeadEdgeLongSeg%, NumNonLeadEdgeLongSeg%, NumFinRadSeg%
   SHARED FinOuterWRTInnerTemp, FinLeadTemp, FinTrailTemp

   FinTemp = FinLeadTemp - ((LeadEdgeLongSegNumber%) / (NumLeadEdgeLongSeg% +
                              NumNonLeadEdgeLongSeg% - 1)) * (FinLeadTemp - FinTrailTemp)
   FinTemp = FinTemp + ((FinRadSegNumber%) / (NumFinRadSeg% - 1)) *
                              FinOuterWRTInnerTemp

END SUB

SUB AssignNoseTemp (ZSegment%, NoseTemp)
```

```
'******************************************************************************************

'THIS SUBROUTINE DETERMINES NOSE CONE TEMPERATURES BY LINEARLY INTERPOLATING
'BETWEEN THE PREVIOUSLY DEFINED TEMPERATURES AT THE FRONT AND REAR OF THE
'NOSE CONE

'******************************************************************************************

   SHARED NumNoseConeLongSeg%, NoseForwTemp, NoseRearTemp

   NoseTemp = NoseRearTemp + (NoseForwTemp - NoseRearTemp) *
                   (((NumNoseConeLongSeg% - 1) - ZSegment%) / (NumNoseConeLongSeg% - 1))

END SUB

SUB CalcAftBodyMidPoint (ZSegment%, FinNumber%, AftBodyRadSegNumber%, XMid, YMid,
                                       ZMid)

'******************************************************************************************

'THIS SUBROUTINE CALCULATED THE MIDPOINT POSITION OF AN AFTBODY FACET BY
'AVERAGING THE POSITIONS OF THE FACET'S FOUR CORNERS

'******************************************************************************************

   SHARED XAftBodyPos(), YAftBodyPos(), ZAftBodyPos()

   XMid = 0
   YMid = 0
   ZMid = 0

   FOR Corner% = 1 TO 4
      XMid = XMid + XAftBodyPos(FinNumber%, AftBodyRadSegNumber%, Corner%) / 4
      YMid = YMid + YAftBodyPos(FinNumber%, AftBodyRadSegNumber%, Corner%) / 4
      ZMid = ZMid + ZAftBodyPos(ZSegment%, Corner%) / 4
      NEXT Corner%

END SUB

SUB CalcAftBodyPos

'******************************************************************************************

'THIS SUBROUTINE CALCULATES THE POSITIONS OF THE CORNERS, THE AREA, AND THE
'COMPONENTS OF THE NORMAL VECTOR FOR THE AFTERBODY FACETS.

'******************************************************************************************

   SHARED NumFins%, ThickFin, DeltaZFin, NumLeadEdgeLongSeg%, RadBody
   SHARED NumNonLeadEdgeLongSeg%, NumAftBodyRadSegPerFin%, LengthNoseCone
   SHARED LengthBody, ZAftBodyPos(), XAftBodyPos(), YAftBodyPos()
   SHARED ZAftBodyNormal, XAftBodyNormal(), YAftBodyNormal(), AftBodyArea

                                                    'the constant cord length of each facet is
                                                    'calculated taking into account the
                                                    'thickness of the fins
   ChordPerAftBodyRadSeg = (2 * 3.14159 * RadBody - NumFins% * ThickFin) / (NumFins% *
                                    NumAftBodyRadSegPerFin%)
                                                    'the constant area of the afterbody facets
                                                    'is calculated
   AftBodyArea = ChordPerAftBodyRadSeg * DeltaZFin
                                                    'each afterbody facet is considered
                                                    'starting at the junction between the body
                                                    'and the afterbody and working rearward to
                                                    'the projectile end
   FOR ZSegment% = 0 TO NumLeadEdgeLongSeg% + NumNonLeadEdgeLongSeg% - 1
```

```
                                                    'ZPosition1 is the more forward facet
                                                    'longitudinal position
    ZPosition1 = LengthNoseCone + LengthBody + ZSegment% * DeltaZFin
                                                    'ZPosition2 is the more rearward facet
                                                    'longitudinal position
    ZPosition2 = LengthNoseCone + LengthBody + (ZSegment% + 1) * DeltaZFin
                                                    'considering the afterbody area between
                                                    'each fin in turn
    FOR FinNumber% = 0 TO NumFins% - 1
                                                    'the radially oriented facets are
                                                    'considered for each longitudinal slice
                                                    'of each afterbody area
      FOR AftBodyRadSegNumber% = 0 TO NumAftBodyRadSegPerFin% - 1
                                                    'the radial angle from the x axis to the
                                                    'afterbody corner positions is calculated
        AngleAftBodyRadSeg1 = (ThickFin * (FinNumber% + .5) + FinNumber% *
                              ChordPerAftBodyRadSeg * NumAftBodyRadSegPerFin% +
                              ChordPerAftBodyRadSeg * AftBodyRadSegNumber%) / RadBody
        AngleAftBodyRadSeg2 = (ThickFin * (FinNumber% + .5) + FinNumber% *
                              ChordPerAftBodyRadSeg * NumAftBodyRadSegPerFin% +
                              ChordPerAftBodyRadSeg * (AftBodyRadSegNumber% + 1)) /
                              RadBody
                                                    'the corner positions are calculated
        ZAftBodyPos(ZSegment%, 1) = ZPosition1
        XAftBodyPos(FinNumber%, AftBodyRadSegNumber%, 1) = -RadBody *
                                  COS(AngleAftBodyRadSeg2)
        YAftBodyPos(FinNumber%, AftBodyRadSegNumber%, 1) = RadBody *
                                  SIN(AngleAftBodyRadSeg2)
        ZAftBodyPos(ZSegment%, 2) = ZPosition2
        XAftBodyPos(FinNumber%, AftBodyRadSegNumber%, 2) = -RadBody *
                                  COS(AngleAftBodyRadSeg2)
        YAftBodyPos(FinNumber%, AftBodyRadSegNumber%, 2) = RadBody *
                                  SIN(AngleAftBodyRadSeg2)
        ZAftBodyPos(ZSegment%, 3) = ZPosition2
        XAftBodyPos(FinNumber%, AftBodyRadSegNumber%, 3) = -RadBody *
                                  COS(AngleAftBodyRadSeg1)
        YAftBodyPos(FinNumber%, AftBodyRadSegNumber%, 3) = RadBody *
                                  SIN(AngleAftBodyRadSeg1)
        ZAftBodyPos(ZSegment%, 4) = ZPosition1
        XAftBodyPos(FinNumber%, AftBodyRadSegNumber%, 4) = -RadBody *
                                  COS(AngleAftBodyRadSeg1)
        YAftBodyPos(FinNumber%, AftBodyRadSegNumber%, 4) = RadBody *
                                  SIN(AngleAftBodyRadSeg1)
                                                    'the components of the normal vectors
                                                    'are calculated
        ZAftBodyNormal = 0
        XAftBodyNormal(FinNumber%, AftBodyRadSegNumber%) =
                                  -COS((AngleAftBodyRadSeg1 + AngleAftBodyRadSeg2) / 2)
        YAftBodyNormal(FinNumber%, AftBodyRadSegNumber%) =
                                  SIN((AngleAftBodyRadSeg1 + AngleAftBodyRadSeg2) / 2)
        NEXT AftBodyRadSegNumber%
      NEXT FinNumber%
    NEXT ZSegment%

END SUB

SUB CalcBodyPos
```

```
'*****************************************************************************************************************

'THIS SUBROUTINE CALCULATES THE POSITIONS OF THE CORNERS, THE AREA, AND THE
'COMPONENTS OF THE NORMAL VECTOR FOR THE BODY FACETS.

'NOTE THAT SYMMETRY ALLOWS THE FOLLOWING SIMPLIFICATIONS:

'  * THE CORD LENGTH OF ALL THE FACETS WILL BE THE SAME

'  * THE Z COMPONENT OF ALL THE FACET BODY NORMALS WILL BE ZERO

'  * ALL THE FACET AREAS WILL BE THE SAME

'*****************************************************************************************************************

   SHARED LengthBody, DeltaZBody, RadBody, NumBodyRadSeg%
   SHARED ZBodyPos(), XBodyPos(), YBodyPos(), BodyArea
   SHARED ZBodyNormal, XBodyNormal(), YBodyNormal(), NumBodyLongSeg%
   SHARED LengthNoseCone
                                                     'constant chord length of each facet is
                                                     'calculated
   MaxBodyChordLength = 2 * 3.14159 * RadBody / NumBodyRadSeg%
                                                     'for the body facets the value of theta
                                                     'is zero.
   TanTheta = 0
   CosTheta = 1
   SinTheta = 0
                                                     'each body facet is considered starting at
                                                     'the junction between the nose and the
                                                     'body and working back towards the
                                                     'junction with the aft body
   FOR ZSegment% = 0 TO NumBodyLongSeg% - 1
                                                     'ZPosition1 is the more forward facet
                                                     'longitudinal position
     ZPosition1 = ZSegment% * DeltaZBody
                                                     'ZPosition2 is the more rearward facet
                                                     'longitudinal position
     ZPosition2 = (ZSegment% + 1) * DeltaZBody
                                                     'the radially oriented facets are
                                                     'considered for each longitudinal slice
     FOR CordSegment% = 0 TO NumBodyRadSeg% - 1
                                                     'the facet corner positions are calculated
       ZBodyPos(ZSegment%, 1) = ZPosition1 + LengthNoseCone
       XBodyPos(CordSegment%, 1) = -RadBody * COS(2 * 3.14159 * (CordSegment% + .5) /
                                   NumBodyRadSeg%)
       YBodyPos(CordSegment%, 1) = RadBody * SIN(2 * 3.14159 * (CordSegment% + .5) /
                                   NumBodyRadSeg%)
       ZBodyPos(ZSegment%, 2) = ZPosition2 + LengthNoseCone
       XBodyPos(CordSegment%, 2) = -RadBody * COS(2 * 3.14159 * (CordSegment% + .5) /
                                   NumBodyRadSeg%)
       YBodyPos(CordSegment%, 2) = RadBody * SIN(2 * 3.14159 * (CordSegment% + .5) /
                                   NumBodyRadSeg%)
       ZBodyPos(ZSegment%, 3) = ZPosition2 + LengthNoseCone
       XBodyPos(CordSegment%, 3) = -RadBody * COS(2 * 3.14159 * (CordSegment% - .5) /
                                   NumBodyRadSeg%)
       YBodyPos(CordSegment%, 3) = RadBody * SIN(2 * 3.14159 * (CordSegment% - .5) /
                                   NumBodyRadSeg%)
       ZBodyPos(ZSegment%, 4) = ZPosition1 + LengthNoseCone
       XBodyPos(CordSegment%, 4) = -RadBody * COS(2 * 3.14159 * (CordSegment% - .5) /
                                   NumBodyRadSeg%)
       YBodyPos(CordSegment%, 4) = RadBody * SIN(2 * 3.14159 * (CordSegment% - .5) /
                                   NumBodyRadSeg%)
                                                     'the facet area is calculated
       BodyArea = MaxBodyChordLength * DeltaZBody
                                                     'the facet normal vector components are
                                                     'calculated
```

```
      ZBodyNormal = 0
      XBodyNormal(CordSegment%) = -CosTheta * COS(2 * 3.14159 * CordSegment% /
                                  NumBodyRadSeg%)
      YBodyNormal(CordSegment%) = CosTheta * SIN(2 * 3.14159 * CordSegment% /
                                  NumBodyRadSeg%)
      NEXT CordSegment%
    NEXT ZSegment%

END SUB

SUB CalcBoxExtremes
'*******************************************************************************

'THIS SUBROUTINE DETERMINES THE EXTREME PROJECTILE POSITIONS SO THAT A
'PROPERLY SIZED GRAPHICS SCREEN CAN BE INITIALIZED

'*******************************************************************************

  SHARED ZNoseConePos(), XNoseConePos(), YNoseConePos(), NumNoseConeLongSeg%
  SHARED NumNoseConeRadSeg%, ZBoxMin, ZBoxMax, XBoxMin, XBoxMax, YBoxMin
  SHARED YBoxMax, ZBodyPos(), XBodyPos(), YBodyPos()
  SHARED NumBodyLongSeg%, NumBodyRadSeg%, ZFinEdgePos(), YFinEdgePos()
  SHARED XFinEdgePos(), NumFins%, NumLeadEdgeLongSeg%
  SHARED NumNonLeadEdgeLongSeg%

  ZBoxMin = 1E+10
  ZBoxMax = -1E+10
  XBoxMin = 1E+10
  XBoxMax = -1E+10
  YBoxMin = 1E+10
  YBoxMax = -1E+10
                                       'checking all the nose cone positions
  FOR ZSegment% = 0 TO NumNoseConeLongSeg% - 1
    FOR CordSegment% = 0 TO NumNoseConeRadSeg% - 1
      FOR Corner% = 1 TO 4
        IF (ZNoseConePos(ZSegment%, Corner%) > ZBoxMax) THEN ZBoxMax =
                        ZNoseConePos(ZSegment%, Corner%)
        IF (ZNoseConePos(ZSegment%, Corner%) < ZBoxMin) THEN ZBoxMin =
                        ZNoseConePos(ZSegment%, Corner%)
        IF (XNoseConePos(ZSegment%, CordSegment%, Corner%) > XBoxMax) THEN
                        XBoxMax = XNoseConePos(ZSegment%, CordSegment%,
                        Corner%)
        IF (XNoseConePos(ZSegment%, CordSegment%, Corner%) < XBoxMin) THEN
                        XBoxMin = XNoseConePos(ZSegment%, CordSegment%, Corner%)
        IF (YNoseConePos(ZSegment%, CordSegment%, Corner%) > YBoxMax) THEN
                        YBoxMax = YNoseConePos(ZSegment%, CordSegment%, Corner%)
        IF (YNoseConePos(ZSegment%, CordSegment%, Corner%) < YBoxMin) THEN
                        YBoxMin = YNoseConePos(ZSegment%, CordSegment%, Corner%)
        NEXT Corner%
      NEXT CordSegment%
    NEXT ZSegment%
                                       'checking all the body positions
  FOR ZSegment% = 0 TO NumBodyLongSeg% - 1
    FOR CordSegment% = 0 TO NumBodyRadSeg% - 1
      FOR Corner% = 1 TO 4
        IF (ZBodyPos(ZSegment%, Corner%) > ZBoxMax) THEN ZBoxMax =
                        ZBodyPos(ZSegment%, Corner%)
        IF (ZBodyPos(ZSegment%, Corner%) < ZBoxMin) THEN ZBoxMin =
                        ZBodyPos(ZSegment%, Corner%)
        IF (XBodyPos(CordSegment%, Corner%) > XBoxMax) THEN XBoxMax =
                        XBodyPos(CordSegment%, Corner%)
        IF (XBodyPos(CordSegment%, Corner%) < XBoxMin) THEN XBoxMin =
                        XBodyPos(CordSegment%, Corner%)
        IF (YBodyPos(CordSegment%, Corner%) > YBoxMax) THEN YBoxMax =
                        YBodyPos(CordSegment%, Corner%)
```

```
            IF (YBodyPos(CordSegment%, Corner%) < YBoxMin) THEN YBoxMin =
                                   YBodyPos(CordSegment%, Corner%)
            NEXT Corner%
         NEXT CordSegment%
      NEXT ZSegment%
                                                 'checking all the fin positions
   FOR FinNumber% = 0 TO NumFins% - 1
      FOR ZSegment% = 0 TO NumLeadEdgeLongSeg% + NumNonLeadEdgeLongSeg% - 1
         FOR Corner% = 1 TO 4
            IF (ZFinEdgePos(ZSegment%, Corner%) > ZBoxMax) THEN ZBoxMax =
                                   ZFinEdgePos(ZSegment%, Corner%)
            IF (ZFinEdgePos(ZSegment%, Corner%) < ZBoxMin) THEN ZBoxMin =
                                   ZFinEdgePos(ZSegment%, Corner%)
            IF (XFinEdgePos(FinNumber%, ZSegment%, Corner%) > XBoxMax) THEN XBoxMax =
                                   XFinEdgePos(FinNumber%, ZSegment%, Corner%)
            IF (XFinEdgePos(FinNumber%, ZSegment%, Corner%) < XBoxMin) THEN XBoxMin =
                                   XFinEdgePos(FinNumber%, ZSegment%, Corner%)
            IF (YFinEdgePos(FinNumber%, ZSegment%, Corner%) > YBoxMax) THEN YBoxMax =
                                   YFinEdgePos(FinNumber%, ZSegment%, Corner%)
            IF (YFinEdgePos(FinNumber%, ZSegment%, Corner%) < YBoxMin) THEN YBoxMin =
                                   YFinEdgePos(FinNumber%, ZSegment%, Corner%)
            NEXT Corner%
         NEXT ZSegment%
      NEXT FinNumber%
                                                 'it is assumed that an aftbody position
                                                 'will never be at an extreme location
END SUB

SUB CalcEulerElem
'*****************************************************************************************

'THIS SUBROUTINE CALCULATES THE MATRIX ELEMENTS THAT ARE NECESSARY TO
'TRANSFORM BETWEEN PROJECTILE AND SPACE COORDINATES

'*****************************************************************************************

   SHARED Euler(), E()

   E(1, 1) = COS(Euler(1)) * COS(Euler(3)) - SIN(Euler(1)) * COS(Euler(2)) * SIN(Euler(3))
   E(1, 2) = -COS(Euler(1)) * SIN(Euler(3)) - SIN(Euler(1)) * COS(Euler(2)) * COS(Euler(3))
   E(1, 3) = SIN(Euler(1)) * SIN(Euler(2))
   E(2, 1) = SIN(Euler(1)) * COS(Euler(3)) + COS(Euler(1)) * COS(Euler(2)) * SIN(Euler(3))
   E(2, 2) = -SIN(Euler(1)) * SIN(Euler(3)) + COS(Euler(1)) * COS(Euler(2)) * COS(Euler(3))
   E(2, 3) = -COS(Euler(1)) * SIN(Euler(2))
   E(3, 1) = SIN(Euler(2)) * SIN(Euler(3))
   E(3, 2) = SIN(Euler(2)) * COS(Euler(3))
   E(3, 3) = COS(Euler(2))

END SUB

SUB CalcFinEdgeMidPoint (FinNumber%, LeadEdgeLongSegNumber%, XMid, YMid, ZMid)
'*****************************************************************************************

'THIS SUBROUTINE CALCULATED THE MIDPOINT POSITIONS OF FIN EDGE FACETS BY
'AVERAGING THE POSITIONS OF THE FACET'S FOUR CORNERS

'*****************************************************************************************

   SHARED XFinEdgePos(), YFinEdgePos(), ZFinEdgePos()

   XMid = 0
   YMid = 0
   ZMid = 0
   FOR Corner% = 1 TO 4
```

```
    XMid = XMid + XFinEdgePos(FinNumber%, LeadEdgeLongSegNumber%, Corner%) / 4
    YMid = YMid + YFinEdgePos(FinNumber%, LeadEdgeLongSegNumber%, Corner%) / 4
    ZMid = ZMid + ZFinEdgePos(LeadEdgeLongSegNumber%, Corner%) / 4
    NEXT Corner%

END SUB

SUB CalcFinPos
'***********************************************************************************************

'THIS SUBROUTINE CALCULATES THE POSITIONS OF THE CORNERS, THE AREA, AND THE
'COMPONENTS OF THE NORMAL VECTOR FOR THE FIN FACETS.

'NOTE THAT SYMMETRY ALLOWS THE FOLLOWING SIMPLIFICATIONS:

'  * ALL THE LEADING EDGE FACETS WILL HAVE THE SAME Z COMPONENT OF THE
'    NORMAL VECTOR

'  * ALL THE LEADING EDGE FACETS HAVE THE SAME AREA

'  * ALL THE NON-LEADING EDGE FACETS HAVE A NORMAL VECTOR Z COMPONENT
'    OF ZERO

'  * ALL THE NON-LEADING EDGE FACETS HAVE THE SAME AREA

'  * ALL THE SIDE FACETS HAVE A NORMAL VECTOR Z COMPONENT OF ZERO

'  * ALL THE SIDE FACETS ON ONE SIDE OF A GIVEN FIN HAVE THE SAME NORMAL
'    VECTOR X AND Y COMPONENTS

'  * ALL THE SIDE FACETS IN THE NON-LEADING EDGE AREA OF THE FIN HAVE THE
'    SAME AREA
'***********************************************************************************************

  SHARED NumFins%, ThickFin, LengthLeadEdgeFin, DeltaZFin
  SHARED NumLeadEdgeLongSeg%, HeightFin, NumNonLeadEdgeLongSeg%
  SHARED DeltaRadFin, NumFinRadSeg%, LengthNoseCone, LengthBody, RadBody
  SHARED ZFinEdgePos(), YFinEdgePos(), XFinEdgePos()
  SHARED ZFinPos(), YFinPos(), XFinPos()
  SHARED ZFinLeadEdgeNormal, XFinLeadEdgeNormal(), YFinLeadEdgeNormal()
  SHARED ZFinNonLeadEdgeNormal, XFinNonLeadEdgeNormal(), YFinNonLeadEdgeNormal()
  SHARED ZFinSideNormal, XFinSideNormal(), YFinSideNormal(), FinLeadEdgeArea
  SHARED FinNonLeadEdgeArea, FinLeadSideArea(), FinNonLeadSideArea
                                                         'the angle that the fin leading edge makes
                                                         'with the projectile body is considered
  FinLeadEdgeSIN = HeightFin / SQR((LengthLeadEdgeFin ^ 2) + (HeightFin ^ 2))
  FinLeadEdgeCOS = LengthLeadEdgeFin / SQR((LengthLeadEdgeFin ^ 2) + (HeightFin ^ 2))
                                                         'the constant area of the fin leading edge
                                                         'facets is calculated
  FinLeadEdgeArea = ThickFin * DeltaZFin / FinLeadEdgeCOS
                                                         'the constant area of the fin non-leading
                                                         'edge edge facets is calculated
  FinNonLeadEdgeArea = ThickFin * DeltaZFin
                                                         'the constant area of the fin side facets
                                                         'that are not associated with a leading
                                                         'edge longitudinal position is calculated
  FinNonLeadSideArea = DeltaZFin * DeltaRadFin
                                                         'each fin is considered in turn
  FOR FinNumber% = 0 TO NumFins% - 1
                                                         'the angle is considered between the x
                                                         'axis and a vector that is normal to the
                                                         'projectile center line and is in the
                                                         'plane of the fin under consideration
    FinRadSIN = SIN(2 * 3.14159 * FinNumber% / NumFins%)
```

```
     FinRadCOS = COS(2 * 3.14159 * FinNumber% / NumFins%)
                                                    'the components of the fins extension
                                                    'away from the central plane of the fin
                                                    'due to the fin's thickness are calculated
     FinThickSINFactor = (ThickFin / 2) * SIN(2 * 3.14159 * FinNumber% / NumFins%)
     FinThickCOSFactor = (ThickFin / 2) * COS(2 * 3.14159 * FinNumber% / NumFins%)
                                                    'the components of the normals of the fin
                                                    'side facets are calculated taking into
                                                    'account the fact that side facets can
                                                    'be paired by which "side" of the fin they
                                                    'are located on.
     ZFinSideNormal = 0
     XFinSideNormal(FinNumber%, 1) = FinRadSIN
     XFinSideNormal(FinNumber%, 2) = -FinRadSIN
     YFinSideNormal(FinNumber%, 1) = FinRadCOS
     YFinSideNormal(FinNumber%, 2) = -FinRadCOS
                                                    'starting at the front of the fin the
                                                    'corner positions, orientations, and areas
                                                    'are calculated for the longitudinal
                                                    'positions corresponding to the leading
                                                    'edge
     FOR LeadEdgeLongSegNumber% = 0 TO NumLeadEdgeLongSeg% - 1
                                                    'the leading edge facets are considered
                                                    'for this leading edge longitudinal
                                                    'position
        ZPosition1 = LengthNoseCone + LengthBody + LeadEdgeLongSegNumber% * DeltaZFin
        ZPosition2 = LengthNoseCone + LengthBody + (LeadEdgeLongSegNumber% + 1) *
                                        DeltaZFin
                                                    'the facet corner positions are calculated
        ZFinEdgePos(LeadEdgeLongSegNumber%, 1) = ZPosition1
        XFinEdgePos(FinNumber%, LeadEdgeLongSegNumber%, 1) = -(RadBody + HeightFin *
                                        LeadEdgeLongSegNumber% / NumLeadEdgeLongSeg%) *
                                        FinRadCOS + FinThickSINFactor
        YFinEdgePos(FinNumber%, LeadEdgeLongSegNumber%, 1) = (RadBody + HeightFin *
                                        LeadEdgeLongSegNumber% / NumLeadEdgeLongSeg%) *
                                        FinRadSIN + FinThickCOSFactor
        ZFinEdgePos(LeadEdgeLongSegNumber%, 2) = ZPosition2
        XFinEdgePos(FinNumber%, LeadEdgeLongSegNumber%, 2) = -(RadBody + HeightFin *
                                        (LeadEdgeLongSegNumber% + 1) / NumLeadEdgeLongSeg%) *
FinRadCOS + FinThickSINFactor
        YFinEdgePos(FinNumber%, LeadEdgeLongSegNumber%, 2) = (RadBody + HeightFin *
                                        (LeadEdgeLongSegNumber% + 1) / NumLeadEdgeLongSeg%) *
FinRadSIN + FinThickCOSFactor
        ZFinEdgePos(LeadEdgeLongSegNumber%, 3) = ZPosition2
        XFinEdgePos(FinNumber%, LeadEdgeLongSegNumber%, 3) = -(RadBody + HeightFin *
                                        (LeadEdgeLongSegNumber% + 1) / NumLeadEdgeLongSeg%) *
FinRadCOS - FinThickSINFactor
        YFinEdgePos(FinNumber%, LeadEdgeLongSegNumber%, 3) = (RadBody + HeightFin *
                                        (LeadEdgeLongSegNumber% + 1) / NumLeadEdgeLongSeg%) *
FinRadSIN - FinThickCOSFactor
        ZFinEdgePos(LeadEdgeLongSegNumber%, 4) = ZPosition1
        XFinEdgePos(FinNumber%, LeadEdgeLongSegNumber%, 4) = -(RadBody + HeightFin *
                                        LeadEdgeLongSegNumber% / NumLeadEdgeLongSeg%) *
                                        FinRadCOS - FinThickSINFactor
        YFinEdgePos(FinNumber%, LeadEdgeLongSegNumber%, 4) = (RadBody + HeightFin *
                                        LeadEdgeLongSegNumber% / NumLeadEdgeLongSeg%) *
                                        FinRadSIN - FinThickCOSFactor
                                                    'the facet normal vector components are
                                                    'calculated
        ZFinLeadEdgeNormal = -FinLeadEdgeSIN
        XFinLeadEdgeNormal(FinNumber%) = -FinLeadEdgeCOS * FinRadCOS
        YFinLeadEdgeNormal(FinNumber%) = FinLeadEdgeCOS * FinRadSIN
                                                    'the facet area is calculated
        FinLeadSideArea(LeadEdgeLongSegNumber%) = .5 * (LeadEdgeLongSegNumber% + 1) *
                                   DeltaZFin * HeightFin * ((LeadEdgeLongSegNumber% + 1) /
                                   NumLeadEdgeLongSeg%) - (.5 * (LeadEdgeLongSegNumber%) *
```

```
                              DeltaZFin * HeightFin * ((LeadEdgeLongSegNumber%) /
                              NumLeadEdgeLongSeg%))
                                                'the side facets are considered for this
                                                'leading edge longitudinal position
        FOR LeadEdgeRadSegNumber% = 0 TO NumFinRadSeg% - 1
                                                'the facet corner positions are calculated
          ZFinPos(LeadEdgeLongSegNumber%, 1) = ZPosition1
          XFinPos(FinNumber%, LeadEdgeLongSegNumber%, LeadEdgeRadSegNumber%, 1, 1)  =
-(RadBody + HeightFin * (LeadEdgeLongSegNumber% /
                         NumLeadEdgeLongSeg%) * (LeadEdgeRadSegNumber% /
                         NumFinRadSeg%)) * FinRadCOS + FinThickSINFactor
          YFinPos(FinNumber%, LeadEdgeLongSegNumber%, LeadEdgeRadSegNumber%, 1, 1)  =
(RadBody + HeightFin * (LeadEdgeLongSegNumber% /                NumLeadEdgeLongSeg%)
* (LeadEdgeRadSegNumber% /                         NumFinRadSeg%)) * FinRadSIN +
FinThickCOSFactor
          ZFinPos(LeadEdgeLongSegNumber%, 2) = ZPosition1
          XFinPos(FinNumber%, LeadEdgeLongSegNumber%, LeadEdgeRadSegNumber%, 1, 2)  =
-(RadBody + HeightFin * (LeadEdgeLongSegNumber% /
                         NumLeadEdgeLongSeg%) * ((LeadEdgeRadSegNumber% + 1) /
                         NumFinRadSeg%)) * FinRadCOS + FinThickSINFactor
          YFinPos(FinNumber%, LeadEdgeLongSegNumber%, LeadEdgeRadSegNumber%, 1, 2)  =
(RadBody + HeightFin * (LeadEdgeLongSegNumber% /                NumLeadEdgeLongSeg%)
* ((LeadEdgeRadSegNumber% + 1) /                     NumFinRadSeg%)) * FinRadSIN +
FinThickCOSFactor
          ZFinPos(LeadEdgeLongSegNumber%, 3) = ZPosition2
          XFinPos(FinNumber%, LeadEdgeLongSegNumber%, LeadEdgeRadSegNumber%, 1, 3)  =
-(RadBody + HeightFin * ((LeadEdgeLongSegNumber% + 1) /
                         NumLeadEdgeLongSeg%) * ((LeadEdgeRadSegNumber% + 1) /
                         NumFinRadSeg%)) * FinRadCOS + FinThickSINFactor
          YFinPos(FinNumber%, LeadEdgeLongSegNumber%, LeadEdgeRadSegNumber%, 1, 3)  =
(RadBody + HeightFin * ((LeadEdgeLongSegNumber% + 1) /
                         NumLeadEdgeLongSeg%) * ((LeadEdgeRadSegNumber% + 1) /
                         NumFinRadSeg%)) * FinRadSIN + FinThickCOSFactor
          ZFinPos(LeadEdgeLongSegNumber%, 4) = ZPosition2
          XFinPos(FinNumber%, LeadEdgeLongSegNumber%, LeadEdgeRadSegNumber%, 1, 4)  =
-(RadBody + HeightFin * ((LeadEdgeLongSegNumber% + 1) /
                         NumLeadEdgeLongSeg%) * (LeadEdgeRadSegNumber% /
                         NumFinRadSeg%)) * FinRadCOS + FinThickSINFactor
          YFinPos(FinNumber%, LeadEdgeLongSegNumber%, LeadEdgeRadSegNumber%, 1, 4)  =
(RadBody + HeightFin * ((LeadEdgeLongSegNumber% + 1) /
                         NumLeadEdgeLongSeg%) * (LeadEdgeRadSegNumber% /
                         NumFinRadSeg%)) * FinRadSIN + FinThickCOSFactor
          XFinPos(FinNumber%, LeadEdgeLongSegNumber%, LeadEdgeRadSegNumber%, 2, 1)  =
-(RadBody + HeightFin * (LeadEdgeLongSegNumber% /
                         NumLeadEdgeLongSeg%) * (LeadEdgeRadSegNumber% /
                         NumFinRadSeg%)) * FinRadCOS - FinThickSINFactor
          YFinPos(FinNumber%, LeadEdgeLongSegNumber%, LeadEdgeRadSegNumber%, 2, 1)  =
(RadBody + HeightFin * (LeadEdgeLongSegNumber% /                NumLeadEdgeLongSeg%)
* (LeadEdgeRadSegNumber% /                         NumFinRadSeg%)) * FinRadSIN -
FinThickCOSFactor
          XFinPos(FinNumber%, LeadEdgeLongSegNumber%, LeadEdgeRadSegNumber%, 2, 2)  =
-(RadBody + HeightFin * (LeadEdgeLongSegNumber% /
                         NumLeadEdgeLongSeg%) * ((LeadEdgeRadSegNumber% + 1) /
                         NumFinRadSeg%)) * FinRadCOS - FinThickSINFactor
          YFinPos(FinNumber%, LeadEdgeLongSegNumber%, LeadEdgeRadSegNumber%, 2, 2)  =
(RadBody + HeightFin * (LeadEdgeLongSegNumber% /                NumLeadEdgeLongSeg%)
* ((LeadEdgeRadSegNumber% + 1) /                     NumFinRadSeg%)) * FinRadSIN -
FinThickCOSFactor
          XFinPos(FinNumber%, LeadEdgeLongSegNumber%, LeadEdgeRadSegNumber%, 2, 3)  =
-(RadBody + HeightFin * ((LeadEdgeLongSegNumber% + 1) /
                         NumLeadEdgeLongSeg%) * ((LeadEdgeRadSegNumber% + 1) /
                         NumFinRadSeg%)) * FinRadCOS - FinThickSINFactor
          YFinPos(FinNumber%, LeadEdgeLongSegNumber%, LeadEdgeRadSegNumber%, 2, 3)  =
(RadBody + HeightFin * ((LeadEdgeLongSegNumber% + 1) /
                         NumLeadEdgeLongSeg%) * ((LeadEdgeRadSegNumber% + 1) /
                         NumFinRadSeg%)) * FinRadSIN - FinThickCOSFactor
```

```
            XFinPos(FinNumber%, LeadEdgeLongSegNumber%, LeadEdgeRadSegNumber%, 2, 4)    =
-(RadBody + HeightFin * ((LeadEdgeLongSegNumber% + 1) /
                            NumLeadEdgeLongSeg%) * (LeadEdgeRadSegNumber% /
                            NumFinRadSeg%)) * FinRadCOS - FinThickSINFactor
            YFinPos(FinNumber%, LeadEdgeLongSegNumber%, LeadEdgeRadSegNumber%, 2, 4)    =
(RadBody + HeightFin * ((LeadEdgeLongSegNumber% + 1) /
                            NumLeadEdgeLongSeg%) * (LeadEdgeRadSegNumber% /
                            NumFinRadSeg%)) * FinRadSIN - FinThickCOSFactor
            NEXT LeadEdgeRadSegNumber%
          NEXT LeadEdgeLongSegNumber%
                                                      'starting at the front of the non-leading
                                                      'edge portion of the fin the corner
                                                      'positions, orientations, and areas are
                                                      'calculated for the longitudinal positions
                                                      'corresponding to the non-leading edge.
        FOR NonLeadEdgeLongSegNumber% = NumLeadEdgeLongSeg% TO
                                      (NumLeadEdgeLongSeg% + NumNonLeadEdgeLongSeg% - 1)
                                                      'the non-leading edge facets are
                                                      'considered for this non-leading edge
                                                      'longitudinal position
          ZPosition1 = LengthNoseCone + LengthBody + NonLeadEdgeLongSegNumber% *
                                      DeltaZFin
          ZPosition2 = LengthNoseCone + LengthBody + (NonLeadEdgeLongSegNumber% + 1) *
                                      DeltaZFin
                                                      'the facet corner positions are calculated
          ZFinEdgePos(NonLeadEdgeLongSegNumber%, 1) = ZPosition1
          XFinEdgePos(FinNumber%, NonLeadEdgeLongSegNumber%, 1) = -(RadBody + HeightFin)    *
FinRadCOS + FinThickSINFactor
          YFinEdgePos(FinNumber%, NonLeadEdgeLongSegNumber%, 1) = (RadBody + HeightFin)    *
FinRadSIN + FinThickCOSFactor
          ZFinEdgePos(NonLeadEdgeLongSegNumber%, 2) = ZPosition2
          XFinEdgePos(FinNumber%, NonLeadEdgeLongSegNumber%, 2) = -(RadBody + HeightFin)    *
FinRadCOS + FinThickSINFactor
          YFinEdgePos(FinNumber%, NonLeadEdgeLongSegNumber%, 2) = (RadBody + HeightFin)    *
FinRadSIN + FinThickCOSFactor
          ZFinEdgePos(NonLeadEdgeLongSegNumber%, 3) = ZPosition2
          XFinEdgePos(FinNumber%, NonLeadEdgeLongSegNumber%, 3) = -(RadBody + HeightFin)    *
FinRadCOS - FinThickSINFactor
          YFinEdgePos(FinNumber%, NonLeadEdgeLongSegNumber%, 3) = (RadBody + HeightFin)    *
FinRadSIN - FinThickCOSFactor
          ZFinEdgePos(NonLeadEdgeLongSegNumber%, 4) = ZPosition1
          XFinEdgePos(FinNumber%, NonLeadEdgeLongSegNumber%, 4) = -(RadBody + HeightFin)    *
FinRadCOS - FinThickSINFactor
          YFinEdgePos(FinNumber%, NonLeadEdgeLongSegNumber%, 4) = (RadBody + HeightFin)    *
FinRadSIN - FinThickCOSFactor
                                                      'the facet normal vector components
                                                      'are calculated
          ZFinNonLeadEdgeNormal = 0
          XFinNonLeadEdgeNormal(FinNumber%) = -FinRadCOS
          YFinNonLeadEdgeNormal(FinNumber%) = FinRadSIN
                                                      'the side facets are considered for this
                                                      'non-leading edge longitudinal position
          FOR NonLeadEdgeRadSegNumber% = 0 TO NumFinRadSeg% - 1
                                                      'the facet corner positions are calculated
            ZFinPos(NonLeadEdgeLongSegNumber%, 1) = ZPosition1
            XFinPos(FinNumber%, NonLeadEdgeLongSegNumber%, NonLeadEdgeRadSegNumber%,
                            1, 1) = -(RadBody + HeightFin * NonLeadEdgeRadSegNumber% /
                            NumFinRadSeg%) * FinRadCOS + FinThickSINFactor
            YFinPos(FinNumber%, NonLeadEdgeLongSegNumber%, NonLeadEdgeRadSegNumber%,
                            1, 1) = (RadBody + HeightFin * NonLeadEdgeRadSegNumber% /
                            NumFinRadSeg%) * FinRadSIN + FinThickCOSFactor
            ZFinPos(NonLeadEdgeLongSegNumber%, 2) = ZPosition1
            XFinPos(FinNumber%, NonLeadEdgeLongSegNumber%, NonLeadEdgeRadSegNumber%,
                            1, 2) = -(RadBody + HeightFin * (NonLeadEdgeRadSegNumber% + 1) /
                            NumFinRadSeg%) * FinRadCOS + FinThickSINFactor
            YFinPos(FinNumber%, NonLeadEdgeLongSegNumber%, NonLeadEdgeRadSegNumber%,
```

```
                    1, 2) = (RadBody + HeightFin * (NonLeadEdgeRadSegNumber% + 1) /
                    NumFinRadSeg%) * FinRadSIN + FinThickCOSFactor
        ZFinPos(NonLeadEdgeLongSegNumber%, 3) = ZPosition2
        XFinPos(FinNumber%, NonLeadEdgeLongSegNumber%, NonLeadEdgeRadSegNumber%,
                    1, 3) = -(RadBody + HeightFin * (NonLeadEdgeRadSegNumber% + 1) /
                    NumFinRadSeg%) * FinRadCOS + FinThickSINFactor
        YFinPos(FinNumber%, NonLeadEdgeLongSegNumber%, NonLeadEdgeRadSegNumber%,
                    1, 3) = (RadBody + HeightFin * (NonLeadEdgeRadSegNumber% + 1) /
                    NumFinRadSeg%) * FinRadSIN + FinThickCOSFactor
        ZFinPos(NonLeadEdgeLongSegNumber%, 4) = ZPosition2
        XFinPos(FinNumber%, NonLeadEdgeLongSegNumber%, NonLeadEdgeRadSegNumber%,
                    1, 4) = -(RadBody + HeightFin * NonLeadEdgeRadSegNumber% /
                    NumFinRadSeg%) * FinRadCOS + FinThickSINFactor
        YFinPos(FinNumber%, NonLeadEdgeLongSegNumber%, NonLeadEdgeRadSegNumber%,
                    1, 4) = (RadBody + HeightFin * NonLeadEdgeRadSegNumber% /
                    NumFinRadSeg%) * FinRadSIN + FinThickCOSFactor
        XFinPos(FinNumber%, NonLeadEdgeLongSegNumber%, NonLeadEdgeRadSegNumber%,
                    2, 1) = -(RadBody + HeightFin * NonLeadEdgeRadSegNumber% /
                    NumFinRadSeg%) * FinRadCOS - FinThickSINFactor
        YFinPos(FinNumber%, NonLeadEdgeLongSegNumber%, NonLeadEdgeRadSegNumber%,
                    2, 1) = (RadBody + HeightFin * NonLeadEdgeRadSegNumber% /
                    NumFinRadSeg%) * FinRadSIN - FinThickCOSFactor
        XFinPos(FinNumber%, NonLeadEdgeLongSegNumber%, NonLeadEdgeRadSegNumber%,
                    2, 2) = -(RadBody + HeightFin * (NonLeadEdgeRadSegNumber% + 1) /
                    NumFinRadSeg%) * FinRadCOS - FinThickSINFactor
        YFinPos(FinNumber%, NonLeadEdgeLongSegNumber%, NonLeadEdgeRadSegNumber%,
                    2, 2) = (RadBody + HeightFin * (NonLeadEdgeRadSegNumber% + 1) /
                    NumFinRadSeg%) * FinRadSIN - FinThickCOSFactor
        XFinPos(FinNumber%, NonLeadEdgeLongSegNumber%, NonLeadEdgeRadSegNumber%,
                    2, 3) = -(RadBody + HeightFin * (NonLeadEdgeRadSegNumber% + 1) /
                    NumFinRadSeg%) * FinRadCOS - FinThickSINFactor
        YFinPos(FinNumber%, NonLeadEdgeLongSegNumber%, NonLeadEdgeRadSegNumber%,
                    2, 3) = (RadBody + HeightFin * (NonLeadEdgeRadSegNumber% + 1) /
                    NumFinRadSeg%) * FinRadSIN - FinThickCOSFactor
        XFinPos(FinNumber%, NonLeadEdgeLongSegNumber%, NonLeadEdgeRadSegNumber%,
                    2, 4) = -(RadBody + HeightFin * NonLeadEdgeRadSegNumber% /
                    NumFinRadSeg%) * FinRadCOS - FinThickSINFactor
        YFinPos(FinNumber%, NonLeadEdgeLongSegNumber%, NonLeadEdgeRadSegNumber%,
                    2, 4) = (RadBody + HeightFin * NonLeadEdgeRadSegNumber% /
                    NumFinRadSeg%) * FinRadSIN - FinThickCOSFactor
        NEXT NonLeadEdgeRadSegNumber%
      NEXT NonLeadEdgeLongSegNumber%
    NEXT FinNumber%

END SUB

SUB CalcFinSideMidPoint (FinNumber%, LeadEdgeLongSegNumber%, FinRadSegNumber%,
SideNumber%, XMid, YMid, ZMid)

'*****************************************************************************************************

'THIS SUBROUTINE CALCULATED THE MIDPOINT POSITIONS OF FIN SIDE FACETS BY
'AVERAGING THE POSITIONS OF THE FACET'S FOUR CORNERS

'*****************************************************************************************************

  SHARED XFinPos(), YFinPos(), ZFinPos()

  XMid = 0
  YMid = 0
  ZMid = 0

  FOR Corner% = 1 TO 4
    XMid = XMid + XFinPos(FinNumber%, LeadEdgeLongSegNumber%, FinRadSegNumber%,
                                  SideNumber%, Corner%) / 4
    YMid = YMid + YFinPos(FinNumber%, LeadEdgeLongSegNumber%, FinRadSegNumber%,
```

```
                                        SideNumber%, Corner%) / 4
    ZMid = ZMid + ZFinPos(LeadEdgeLongSegNumber%, Corner%) / 4
    NEXT Corner%

END SUB

SUB CalcNoseConePos

'*************************************************************************************************************

'THIS SUBROUTINE CALCULATES THE POSITIONS OF THE CORNERS, THE AREA, AND THE
'COMPONENTS OF THE NORMAL VECTOR FOR THE NOSE CONE FACETS.

'NOTE THAT SYMMETRY ALLOWS THE FOLLOWING SIMPLIFICATION:

'  * THE Z COMPONENT OF ALL THE FACET NORMAL VECTORS WILL BE THE SAME

'*************************************************************************************************************

  SHARED LengthNoseCone, DeltaZNoseCone, NumNoseConeLongSeg%, RadNoseCone
  SHARED NumNoseConeRadSeg%, ZNoseConePos(), XNoseConePos(), YNoseConePos()
  SHARED NoseConeArea(), ZNoseConeNormal, XNoseConeNormal()
  SHARED YNoseConeNormal()
                                                'the radial segment cord is calculated at
                                                'the base (or largest diameter) of the
                                                'nose cone
  MaxNoseConeChordLength = 2 * 3.14159 * RadNoseCone / NumNoseConeRadSeg%
                                                'theta is the angle between the center
                                                'line of the projectile and the surface
                                                'of the nose cone
  TanTheta = RadNoseCone / LengthNoseCone
  CosTheta = LengthNoseCone / SQR((RadNoseCone ^ 2) + (LengthNoseCone ^ 2))
  SinTheta = RadNoseCone / SQR((RadNoseCone ^ 2) + (LengthNoseCone ^ 2))
                                                'each nose cone facet is considered
                                                'starting at the tip and working towards
                                                'the base
  FOR ZSegment% = 0 TO NumNoseConeLongSeg% - 1
                                                'ZPosition1 is the more forward facet
                                                'longitudinal position
    ZPosition1 = ZSegment% * DeltaZNoseCone
                                                'ZPosition2 is the more rearward facet
                                                'longitudinal position
    ZPosition2 = (ZSegment% + 1) * DeltaZNoseCone
                                                'the radially oriented facets are
                                                'considered for each longitudinal slice
    FOR CordSegment% = 0 TO NumNoseConeRadSeg% - 1
                                                'the facet corner positions are calculated
      ZNoseConePos(ZSegment%, 1) = ZPosition1
      XNoseConePos(ZSegment%, CordSegment%, 1) = -ZPosition1 * TanTheta * COS(2 *
                               3.14159 * (CordSegment% + .5) / NumNoseConeRadSeg%)
      YNoseConePos(ZSegment%, CordSegment%, 1) = ZPosition1 * TanTheta * SIN(2 *
                               3.14159 * (CordSegment% + .5) / NumNoseConeRadSeg%)
      ZNoseConePos(ZSegment%, 2) = ZPosition2
      XNoseConePos(ZSegment%, CordSegment%, 2) = -ZPosition2 * TanTheta * COS(2 *
                               3.14159 * (CordSegment% + .5) / NumNoseConeRadSeg%)
      YNoseConePos(ZSegment%, CordSegment%, 2) = ZPosition2 * TanTheta * SIN(2 *
                               3.14159 * (CordSegment% + .5) / NumNoseConeRadSeg%)
      ZNoseConePos(ZSegment%, 3) = ZPosition2
      XNoseConePos(ZSegment%, CordSegment%, 3) = -ZPosition2 * TanTheta * COS(2 *
                               3.14159 * (CordSegment% - .5) / NumNoseConeRadSeg%)
      YNoseConePos(ZSegment%, CordSegment%, 3) = ZPosition2 * TanTheta * SIN(2 *
                               3.14159 * (CordSegment% - .5) / NumNoseConeRadSeg%)
      ZNoseConePos(ZSegment%, 4) = ZPosition1
      XNoseConePos(ZSegment%, CordSegment%, 4) = -ZPosition1 * TanTheta * COS(2 *
                               3.14159 * (CordSegment% - .5) / NumNoseConeRadSeg%)
      YNoseConePos(ZSegment%, CordSegment%, 4) = ZPosition1 * TanTheta * SIN(2 *
```

```
                          3.14159 * (CordSegment% - .5) / NumNoseConeRadSeg%)
                                                'the facet areas are calculated
     NoseConeArea(ZSegment%) = MaxNoseConeChordLength * ((ZPosition1 + ZPosition2) /
                          2) * DeltaZNoseCone / (LengthNoseCone * CosTheta)
                                                'the facet normal vector components are
                                                'calculated
     ZNoseConeNormal = -SinTheta
     XNoseConeNormal(CordSegment%) = -CosTheta * COS(2 * 3.14159 * CordSegment% /
                          NumNoseConeRadSeg%)
     YNoseConeNormal(CordSegment%) = CosTheta * SIN(2 * 3.14159 * CordSegment% /
                          NumNoseConeRadSeg%)
     NEXT CordSegment%
   NEXT ZSegment%

END SUB

SUB CalcScreenSize
'**************************************************************************************************************

'THIS SUBROUTINE CALCULATES AN APPROPRIATE SCALE FOR THE GRAPHICS DISPLAY
'WINDOW. IT OPERATES BY FIRST TRANSFORMING THE PREVIOUSLY DETERMINED
'PROJECTILE EXTREME POSITIONS FROM BODY TO SPACE COORDINATES. CALCULATIONS
'ARE THEN PERFORMED TO DETERMINE WHETHER THE GRAPHIC IMAGE SIZE IS
CONSTRAINED
'BY THE SIZE OF THE GRAPHIC WINDOW IN THE HORIZONTAL OR VERTICAL DIRECTION.
'AN APPROPRIATE SCALE FACTOR IS THEN COMPUTED.
'**************************************************************************************************************

  SHARED XBoxMin, XBoxMax, YBoxMin, YBoxMax, ZBoxMin, ZBoxMax, Scale
  SHARED Control$, ZSpaceMin, ZSpaceMax, XSpaceMin, XSpaceMax, YSpaceMin
  SHARED YSpaceMax, ScreenWidth, ScreenHeight

  DIM ZM(2), YM(2), XM(2)
                                                'projectile extreme position values are
                                                'transferred to array variables to
                                                'facilitate computation.

  ZM(1) = ZBoxMin
  ZM(2) = ZBoxMax
  YM(1) = YBoxMin
  YM(2) = YBoxMax
  XM(1) = XBoxMin
  XM(2) = XBoxMax

  ZSpaceMin = 9999
  ZSpaceMax = -9999

  YSpaceMin = 9999
  YSpaceMax = -9999

  XSpaceMin = 9999
  XSpaceMax = -9999
                                                'extreme projectile positions in the body
                                                'coordinate system are transformed to
                                                'the viewer, or space, coordinate system.

  FOR K% = 1 TO 2
    FOR J% = 1 TO 2
      FOR I% = 1 TO 2
        CALL TransCoordBS(XM(I%), YM(J%), ZM(K%), XSpace, YSpace, ZSpace)
        IF (XSpace > XSpaceMax) THEN XSpaceMax = XSpace
        IF (XSpace < XSpaceMin) THEN XSpaceMin = XSpace
        IF (YSpace > YSpaceMax) THEN YSpaceMax = YSpace
        IF (YSpace < YSpaceMin) THEN YSpaceMin = YSpace
        IF (ZSpace > ZSpaceMax) THEN ZSpaceMax = ZSpace
        IF (ZSpace < ZSpaceMin) THEN ZSpaceMin = ZSpace
```

```
            NEXT I%
         NEXT J%
      NEXT K%
                                                      'the ratio of the space frame extreme
                                                      'positions in the z and y directions are
                                                      'compared to the ratio of the graphic
                                                      'screen's dimensions in the horizontal
                                                      'and vertical directions to determine the
                                                      'more restrictive dimension. a graphic
                                                      'scale factor is then calculated.
   IF (ABS(ZSpaceMax - ZSpaceMin) / ScreenWidth < ABS(YSpaceMax - YSpaceMin) /
                                             ScreenHeight) THEN
      Scale = ScreenHeight / ABS(YSpaceMax - YSpaceMin)
      Control$ = "Y"
      END IF

   IF (ABS(ZSpaceMax - ZSpaceMin) / ScreenWidth >= ABS(YSpaceMax - YSpaceMin) /
                                             ScreenHeight) THEN
      Scale = ScreenWidth / ABS(ZSpaceMax - ZSpaceMin)
      Control$ = "Z"
      END IF

END SUB

SUB Eclipse (XTest, YTest, ZTest, CurrentFinNumber%, FinUnitCOSValue, Type$, Ans%)
'**********************************************************************************

'THIS SUBROUTINE DETERMINES WHETHER THE VIEW OF A GIVEN FACET WILL BE BLOCKED
'BY ANY OTHER FACET. A NUMBER OF SYMMETRY BASED ASSUMPTIONS ARE USED TO
'SIMPLIFY THIS PROCEDURE.

'  1) THE UNIT NORMAL VECTOR OF ALL THE FACETS CHECKED BY THIS ROUTINE MAKE
'     AN ANGLE WITH THE UNIT VECTOR TOWARDS THE VIEWER THAT HAS A POSITIVE
'     COSINE

'  2) NOSE CONE AND BODY FACETS DO NOT REQUIRE THIS TYPE OF TESTING

'  3) THE NOSE CONE AND BODY FACETS ARE ALWAYS CLOSER TO THE VIEWER THAN
'     THE FIN FACETS

'  4) THE NOSE CONE, BODY, AND AFTBODY WILL NEVER BLOCK A FACET ON A FIN FOR
'     WHICH THE UNIT VECTOR THAT IS IN THE PLANE OF THE FIN AND PERPENDICULAR
'     TO THE LONGITUDINAL BODY AXIS MAKES AN ANGLE WITH THE UNIT VECTOR
'     TOWARDS THE VIEWER THAT HAS A POSITIVE COSINE

'  5) THE NOSE CONE AND BODY WILL NEVER BLOCK AN AFTBODY FACET

'  6) A FACET OF A GIVEN FIN WILL NEVER BE BLOCKED BY ANOTHER FACET IN THAT
'     FIN

'  7) AN AFTBODY FACET CAN ONLY BE BLOCKED BY A FACET ON A FIN FOR WHICH THE
'     DOT PRODUCT BETWEEN THE UNIT VECTOR THAT IS IN THE PLANE OF THE FIN AND
'     PERPENDICULAR TO THE LONGITUDINAL BODY AXIS AND THE UNIT VECTOR TOWARD
'     THE VIEWER HAS A NON-NEGATIVE VALUE

'  8) A FACET ON A FIN CAN ONLY BE BLOCKED BY A FACET ON ANOTHER FIN FOR WHICH
'     THE DOT PRODUCT BETWEEN THE UNIT VECTOR THAT IS IN THE PLANE OF THE FIN
'     AND PERPENDICULAR TO THE LONGITUDINAL BODY AXIS AND THE UNIT VECTOR
'     TOWARDS THE VIEWER HAS A VALUE CLOSER TO ONE.

'  9) AN AFTBODY FACET WILL NEVER BE BLOCKED BY ANOTHER AFTBODY FACET

' 10) AN AFTBODY FACET WILL NEVER BLOCK A FIN FACET ON A FIN FOR WHICH THE
'     DOT PRODUCT BETWEEN THE UNIT VECTOR THAT IS IN THE PLANE OF THE FIN AND
'     PERPENDICULAR TO THE LONGITUDINAL BODY AXIS AND THE UNIT VECTOR TOWARD
```

```
'    THE VIEWER HAS A NON-NEGATIVE VALUE
'**************************************************************************************************************************

     SHARED NumNoseConeLongSeg%, NumNoseConeRadSeg%, XNoseConePos()
     SHARED YNoseConePos(), ZNoseConePos(), XNoseConeNormal(), YNoseConeNormal()
     SHARED ZNoseConeNormal, XBodyNormal(), YBodyNormal(), ZBodyNormal
     SHARED NumBodyLongSeg%, NumBodyRadSeg%, NumFinRadSeg%
     SHARED XBodyPos(), YBodyPos(), ZBodyPos(), NumFins%
     SHARED NumLeadEdgeLongSeg%, NumNonLeadEdgeLongSeg%
     SHARED XFinLeadEdgeNormal(), XViewNormal, YFinLeadEdgeNormal()
     SHARED YViewNormal, ZFinLeadEdgeNormal, ZViewNormal
     SHARED XFinNonLeadEdgeNormal(), YFinNonLeadEdgeNormal()
     SHARED ZFinNonLeadEdgeNormal, XFinEdgePos(), YFinEdgePos(), ZFinEdgePos()
     SHARED XFinSideNormal(), YFinSideNormal(), ZFinSideNormal
     SHARED XFinPos(), YFinPos(), ZFinPos(), NumAftBodyRadSegPerFin%
     SHARED XAftBodyPos(), YAftBodyPos(), ZAftBodyPos()
     SHARED XAftBodyNormal(), YAftBodyNormal(), ZAftBodyNormal, Scale
                                                   'as each projectile facet is considered
                                                   'its position is denoted by a dot. the
                                                   'following parameters determine the
                                                   'appearance of this dot
     NumBlinks% = 1
     BlinkDuration = 1
     NewColor = 15
                                                   'initial state set to nonhidden condition
     Ans% = 1
                                                   'simplification #4 is tested
     IF ((FinUnitCOSValue > 0) AND (Type$ = "Fin")) THEN GOTO SkipNoseConeAndBody

                                                   'simplification #5 is tested

     IF (Type$ = "AftBody") THEN GOTO SkipNoseConeAndBody
                                                   'checking for blocking by nose cone
                                                   'facets

                                                   'each nose cone transverse segment is
                                                   'considered
     FOR ZSegment% = 0 TO NumNoseConeLongSeg% - 1
                                                   'each radial facet within the transverse
                                                   'segment is considered
        FOR CordSegment% = 0 TO NumNoseConeRadSeg% - 1
                                                   'only those nose cone facets that have a
                                                   'normal vector that points "towards" the
                                                   'viewer are considered
           NormalDotProduct = XNoseConeNormal(CordSegment%) * XViewNormal +
                              YNoseConeNormal(CordSegment%) * YViewNormal +
                              ZNoseConeNormal * ZViewNormal
           IF (NormalDotProduct > 0) THEN
                                                   'the midpoint position of the nose cone
                                                   'facet under consideration is calculated
              ZNow = 0
              YNow = 0
              FOR Corner% = 1 TO 4
                 CALL TransCoordBS(XNoseConePos(ZSegment%, CordSegment%, Corner%),
                       YNoseConePos(ZSegment%, CordSegment%, Corner%),
                       ZNoseConePos(ZSegment%, Corner%), XSpace, YSpace, ZSpace)
                 ZNow = ZNow + ZSpace / 4
                 YNow = YNow + YSpace / 4
                 NEXT Corner%
                                                   'the original color of the midpoint
                                                   'position pixel is determined and saved
              OldColor = POINT(ZNow * Scale, YNow * Scale)
                                                   'the midpoint position pixel of the nose
                                                   'cone facet under consideration is flashed
              FOR I% = 1 TO NumBlinks%
```

```
   CALL Plot3DPoint(1, YNow, ZNow, OldColor)
   FOR T = 1 TO BlinkDuration: NEXT T
   CALL Plot3DPoint(1, YNow, ZNow, NewColor)
   FOR T = 1 TO BlinkDuration: NEXT T
   NEXT I%
                                             'the vertical extremes of the possible
                                             'blocking nose cone facet are determined
YFacetMin = 9999
YFacetMax = -9999
FOR Corner% = 1 TO 4
   CALL TransCoordBS(XNoseConePos(ZSegment%, CordSegment%, Corner%),
            YNoseConePos(ZSegment%, CordSegment%, Corner%),
            ZNoseConePos(ZSegment%, Corner%), XSpace, YSpace, ZSpace)
   IF (YSpace > YFacetMax) THEN YFacetMax = YSpace
   IF (YSpace < YFacetMin) THEN YFacetMin = YSpace
   NEXT Corner%
                                             'the midpoint vertical position of the
                                             'facet that is being tested for blockage
                                             'is compared to the vertical extremes of
                                             'the potentially blocking nose cone facet.
                                             'if the midpoint is contained within the
                                             'range of the nose cone facet's extremes,
                                             'then further testing will be performed
                                             'that considers horizontal overlap. if
                                             'not, then this nose cone facet can not
                                             'be blocking.
IF ((YFacetMin < YTest) AND (YFacetMax > YTest)) THEN
                                             'the horizontal extremes of the possible
                                             'blocking nose cone facet are determined
   ZFacetMin = 9999
   ZFacetMax = -9999
   FOR Corner% = 1 TO 4
      CALL TransCoordBS(XNoseConePos(ZSegment%, CordSegment%, Corner%),
            YNoseConePos(ZSegment%, CordSegment%, Corner%),
            ZNoseConePos(ZSegment%, Corner%), XSpace, YSpace, ZSpace)
      IF (ZSpace > ZFacetMax) THEN ZFacetMax = ZSpace
      IF (ZSpace < ZFacetMin) THEN ZFacetMin = ZSpace
      NEXT Corner%
                                             'the midpoint horizontal position of the
                                             'facet that is being tested for blockage
                                             'is compared to the horizontal extremes of
                                             'the potentially blocking nose cone facet.
                                             'if the midpoint is contained within the
                                             'range of the nose cone facet's extremes,
                                             'then it is concluded that the nose cone
                                             'facet will block the facet being tested.
   IF ((ZFacetMin < ZTest) AND (ZFacetMax > ZTest)) THEN
      Ans% = 0
      BEEP
                                             'the nose cone facet midpoint position
                                             'pixel is returned to its original color
      CALL Plot3DPoint(1, YNow, ZNow, OldColor)
      GOTO Hidden
                                             'if this nose cone facet is determined not
                                             'to block the facet under consideration,
                                             'then the nose cone facet's midpoint
                                             'position pixel is simply returned to its
                                             'original color
      ELSE
      CALL Plot3DPoint(1, YNow, ZNow, OldColor)
      END IF
                                             'if this nose cone facet is determined not
                                             'to block the facet under consideration,
                                             'then the nose cone facet's midpoint
                                             'position pixel is simply returned to its
                                             'original color
```

```
            ELSE
            CALL Plot3DPoint(1, YNow, ZNow, OldColor)
            END IF
         END IF
      NEXT CordSegment%
   NEXT ZSegment%
                                          'checking for blocking by body facets

                                          'each body transverse segment is
                                          'considered
FOR ZSegment% = 0 TO NumBodyLongSeg% - 1
                                          'each radial facet within the transverse
                                          'segment is considered
   FOR CordSegment% = 0 TO NumBodyRadSeg% - 1
                                          'only those body facets that have a normal
                                          'vector that points "towards" the viewer
                                          'are considered
      NormalDotProduct = XBodyNormal(CordSegment%) * XViewNormal +
                         YBodyNormal(CordSegment%) * YViewNormal +
                         ZBodyNormal * ZViewNormal
      IF (NormalDotProduct > 0) THEN
                                          'the midpoint position of the body facet
                                          'under consideration is calculated
         ZNow = 0
         YNow = 0
         FOR Corner% = 1 TO 4
            CALL TransCoordBS(XBodyPos(CordSegment%, Corner%),
                         YBodyPos(CordSegment%, Corner%), ZBodyPos(ZSegment%,
                         Corner%), XSpace, YSpace, ZSpace)
            ZNow = ZNow + ZSpace / 4
            YNow = YNow + YSpace / 4
            NEXT Corner%
                                          'the original color of the midpoint
                                          'position pixel is determined and saved
         OldColor = POINT(ZNow * Scale, YNow * Scale)
                                          'the midpoint position pixel of the body
                                          'facet under consideration is flashed
         FOR I% = 1 TO NumBlinks%
            CALL Plot3DPoint(1, YNow, ZNow, OldColor)
            FOR T = 1 TO BlinkDuration: NEXT T
            CALL Plot3DPoint(1, YNow, ZNow, NewColor)
            FOR T = 1 TO BlinkDuration: NEXT T
            NEXT I%
                                          'the vertical extremes of the possible
                                          'blocking body facet are determined
         YFacetMin = 9999
         YFacetMax = -9999
         FOR Corner% = 1 TO 4
            CALL TransCoordBS(XBodyPos(CordSegment%, Corner%),
                         YBodyPos(CordSegment%, Corner%), ZBodyPos(ZSegment%,
                         Corner%), XSpace, YSpace, ZSpace)
            IF (YSpace > YFacetMax) THEN YFacetMax = YSpace
            IF (YSpace < YFacetMin) THEN YFacetMin = YSpace
            NEXT Corner%
                                          'the midpoint vertical position of the
                                          'facet that is being tested for blockage
                                          'is compared to the vertical extremes of
                                          'the potentially blocking body facet. if
                                          'the midpoint is contained within the
                                          'range of the body facet's extremes, then
                                          'further testing will be performed that
                                          'considers horizontal overlap. if
                                          'not, then this body facet can not be
                                          'blocking.
         IF ((YFacetMin < YTest) AND (YFacetMax > YTest)) THEN
                                          'the horizontal extremes of the possible
```

```
                                                  'blocking body facet are determined
          ZFacetMin = 9999
          ZFacetMax = -9999
          FOR Corner% = 1 TO 4
            CALL TransCoordBS(XBodyPos(CordSegment%, Corner%),
                            YBodyPos(CordSegment%, Corner%), ZBodyPos(ZSegment%,
                            Corner%), XSpace, YSpace, ZSpace)
            IF (ZSpace > ZFacetMax) THEN ZFacetMax = ZSpace
            IF (ZSpace < ZFacetMin) THEN ZFacetMin = ZSpace
            NEXT Corner%
                                                  'the midpoint horizontal position of the
                                                  'facet that is being tested for blockage
                                                  'is compared to the horizontal extremes of
                                                  'the potentially blocking body facet. if
                                                  'the midpoint is contained within the
                                                  'range of the body facet's extremes, then
                                                  'it is concluded that the body facet will
                                                  'block the facet being tested.
          IF ((ZFacetMin < ZTest) AND (ZFacetMax > ZTest)) THEN
            Ans% = 0
            BEEP
                                                  'the body facet midpoint position pixel
                                                  'is returned to its original color
            CALL Plot3DPoint(1, YNow, ZNow, OldColor)
            GOTO Hidden
                                                  'if this body facet is determined not to
                                                  'block the facet under consideration,
                                                  'then the body facet's midpoint position
                                                  'pixel is simply returned to its original
                                                  'color
            ELSE
            CALL Plot3DPoint(1, YNow, ZNow, OldColor)
            END IF
                                                  'if this body facet is determined not to
                                                  'block the facet under consideration,
                                                  'then the body facet's midpoint position
                                                  'pixel is simply returned to its original
                                                  'color
          ELSE
          CALL Plot3DPoint(1, YNow, ZNow, OldColor)
          END IF
        END IF
      NEXT CordSegment%
    NEXT ZSegment%

SkipNoseConeAndBody:
                                                  'checking for blocking by fin facets

                                                  'each fin is considered
  FOR FinNumber% = 0 TO NumFins% - 1
                                                  'simplification #6 is tested
    IF ((FinNumber% = CurrentFinNumber%) AND (Type$ = "Fin")) THEN GOTO SkipFin
                                                  'the x and y components in the body
                                                  'coordinate system are determined for a
                                                  'unit vector in the fin under
                                                  'consideration that is normal to the
                                                  'longitudinal axis
    TestFinUnitRadXVector = -COS(2 * 3.14159 * FinNumber% / NumFins%)
    TestFinUnitRadYVector = SIN(2 * 3.14159 * FinNumber% / NumFins%)

                                                  'this unit vector is dotted with the unit
                                                  'vector pointing towards the viewer to
                                                  'determine whether the unit vector in the
                                                  'fin points towards or away from the
                                                  'viewer
    TestFinUnitCOSValue = TestFinUnitRadXVector * XViewNormal + TestFinUnitRadYVector *
```

```
                                  YViewNormal
                                                  'simplification #7 is tested
    IF ((TestFinUnitCOSValue < 0) AND (Type$ = "AftBody")) THEN GOTO SkipFin
                                                  'simplification #8 is tested
    IF ((TestFinUnitCOSValue < FinUnitCOSValue) AND (Type$ = "Fin")) THEN GOTO SkipFin
                                                  'each longitudinal segment of the fin is
                                                  'considered
    FOR LeadEdgeLongSegNumber% = 0 TO NumLeadEdgeLongSeg% +
                                  NumNonLeadEdgeLongSeg% - 1
                                                  'each radial segment of the longitudinal
                                                  'segment is considered
      FOR FinRadSegNumber% = 0 TO NumFinRadSeg% - 1
                                                  'both sides of the radial segment are
                                                  'considered
        FOR SideNumber% = 1 TO 2
                                                  'only those fin facets that have a normal
                                                  'vector that points "towards" the viewer
                                                  'are considered
          NormalDotProduct = XFinSideNormal(FinNumber%, SideNumber%) * XViewNormal +
YFinSideNormal(FinNumber%, SideNumber%) * YViewNormal
          IF (NormalDotProduct > 0) THEN
                                                  'the midpoint position of the fin facet
                                                  'under consideration is calculated
            ZNow = 0
            YNow = 0
            FOR Corner% = 1 TO 4
              CALL TransCoordBS(XFinPos(FinNumber%, LeadEdgeLongSegNumber%,
                     FinRadSegNumber%, SideNumber%, Corner%),
                     YFinPos(FinNumber%, LeadEdgeLongSegNumber%,
                     FinRadSegNumber%, SideNumber%, Corner%),
                     ZFinPos(LeadEdgeLongSegNumber%, Corner%), XSpace, YSpace,
                     ZSpace)
              ZNow = ZNow + ZSpace / 4
              YNow = YNow + YSpace / 4
              NEXT Corner%
                                                  'the original color of the midpoint
                                                  'position pixel is determined and saved
            OldColor = POINT(ZNow * Scale, YNow * Scale)
                                                  'the midpoint position pixel of the fin
                                                  'facet under consideration is flashed
            FOR I% = 1 TO NumBlinks%
              CALL Plot3DPoint(1, YNow, ZNow, OldColor)
              FOR T = 1 TO BlinkDuration: NEXT T
              CALL Plot3DPoint(1, YNow, ZNow, NewColor)
              FOR T = 1 TO BlinkDuration: NEXT T
              NEXT I%
                                                  'the farthest distance of the fin facet
                                                  'from the viewer is determined
            XFacetMax = -9999
            FOR Corner% = 1 TO 4
              CALL TransCoordBS(XFinPos(FinNumber%, LeadEdgeLongSegNumber%,
                     FinRadSegNumber%, SideNumber%, Corner%),
                     YFinPos(FinNumber%, LeadEdgeLongSegNumber%,
                     FinRadSegNumber%, SideNumber%, Corner%),
                     ZFinPos(LeadEdgeLongSegNumber%, Corner%), XSpace, YSpace,
                     ZSpace)
              IF (XSpace > XFacetMax) THEN XFacetMax = XSpace
              NEXT Corner%
                                                  'further testing continues only if the
                                                  'farthest distance of the fin facet from
                                                  'the viewer is less than the distance to
                                                  'the projectile facet under consideration
            IF (XFacetMax < XTest) THEN
                                                  'the vertical extremes of the possible
                                                  'blocking fin facet are determined
              YFacetMin = 9999
```

```
YFacetMax = -9999
FOR Corner% = 1 TO 4
   CALL TransCoordBS(XFinPos(FinNumber%, LeadEdgeLongSegNumber%,
                  FinRadSegNumber%, SideNumber%, Corner%),
                  YFinPos(FinNumber%, LeadEdgeLongSegNumber%,
                  FinRadSegNumber%, SideNumber%, Corner%), ZFinPos(
                  LeadEdgeLongSegNumber%, Corner%), XSpace, YSpace,
                  ZSpace)
   IF (YSpace > YFacetMax) THEN YFacetMax = YSpace
   IF (YSpace < YFacetMin) THEN YFacetMin = YSpace
   NEXT Corner%
                                    'the midpoint vertical position of the
                                    'facet that is being tested for blockage
                                    'is compared to the vertical extremes of
                                    'the potentially blocking fin facet. if
                                    'the midpoint is contained within the
                                    'range of the fin facet's extremes, then
                                    'further testing will be performed that
                                    'considers horizontal overlap. if
                                    'not, then this fin facet can not be
                                    'blocking.
IF ((YFacetMin < YTest) AND (YFacetMax > YTest)) THEN
                                    'the horizontal extremes of the possible
                                    'blocking fin facet are determined
   ZFacetMin = 9999
   ZFacetMax = -9999
   FOR Corner% = 1 TO 4
      CALL TransCoordBS(XFinPos(FinNumber%, LeadEdgeLongSegNumber%,
                  FinRadSegNumber%, SideNumber%, Corner%),
                  YFinPos(FinNumber%, LeadEdgeLongSegNumber%,
                  FinRadSegNumber%, SideNumber%, Corner%), ZFinPos(
                  LeadEdgeLongSegNumber%, Corner%), XSpace, YSpace,
                  ZSpace)
      IF (ZSpace > ZFacetMax) THEN ZFacetMax = ZSpace
      IF (ZSpace < ZFacetMin) THEN ZFacetMin = ZSpace
      NEXT Corner%
                                    'the midpoint horizontal position of the
                                    'facet that is being tested for blockage
                                    'is compared to the horizontal extremes of
                                    'the potentially blocking fin facet. if
                                    'the midpoint is contained within the
                                    'range of the fin facet's extremes, then
                                    'it is concluded that the fin facet will
                                    'block the facet being tested.
   IF ((ZFacetMin < ZTest) AND (ZFacetMax > ZTest)) THEN
         BEEP
                                    'the fin facet midpoint position pixel
                                    'is returned to its original color
      CALL Plot3DPoint(1, YNow, ZNow, OldColor)
      Ans% = 0
      GOTO Hidden
                                    'if this fin facet is determined not to
                                    'block the facet under consideration,
                                    'then the fin facet's midpoint position
                                    'pixel is simply returned to its original
                                    'color
      ELSE
      CALL Plot3DPoint(1, YNow, ZNow, OldColor)
      END IF
                                    'if this fin facet is determined not to
                                    'block the facet under consideration,
                                    'then the fin facet's midpoint position
                                    'pixel is simply returned to its original
                                    'color
   ELSE
   CALL Plot3DPoint(1, YNow, ZNow, OldColor)
```

```
            END IF
                                          'if this fin facet is determined not to
                                          'block the facet under consideration,
                                          'then the fin facet's midpoint position
                                          'pixel is simply returned to its original
                                          'color
          ELSE
          CALL Plot3DPoint(1, YNow, ZNow, OldColor)
          END IF
        END IF
      NEXT SideNumber%
    NEXT FinRadSegNumber%
                                          'the leading edge of each longitudinal
                                          'segment of the fin is considered
  IF (LeadEdgeLongSegNumber% < NumLeadEdgeLongSeg%) THEN
    NormalDotProduct = XFinLeadEdgeNormal(FinNumber%) * XViewNormal +
                       YFinLeadEdgeNormal(FinNumber%) * YViewNormal +
                       ZFinLeadEdgeNormal * ZViewNormal
    ELSE
    NormalDotProduct = XFinNonLeadEdgeNormal(FinNumber%) * XViewNormal +
                       YFinNonLeadEdgeNormal(FinNumber%) * YViewNormal
    END IF
                                          'only those fin facets that have a normal
                                          'vector that points "towards" the viewer
                                          'are considered
  IF (NormalDotProduct > 0) THEN
                                          'the midpoint position of the fin facet
                                          'under consideration is calculated
    ZNow = 0
    YNow = 0
    FOR Corner% = 1 TO 4
      CALL TransCoordBS(XFinEdgePos(FinNumber%, LeadEdgeLongSegNumber%,
                       Corner%), YFinEdgePos(FinNumber%,
                       LeadEdgeLongSegNumber%, Corner%),
                       ZFinEdgePos(LeadEdgeLongSegNumber%, Corner%), XSpace,
                       YSpace, ZSpace)
      ZNow = ZNow + ZSpace / 4
      YNow = YNow + YSpace / 4
      NEXT Corner%
                                          'the original color of the midpoint
                                          'position pixel is determined and saved
    OldColor = POINT(ZNow * Scale, YNow * Scale)
                                          'the midpoint position pixel of the fin
                                          'facet under consideration is flashed
    FOR I% = 1 TO NumBlinks%
      CALL Plot3DPoint(1, YNow, ZNow, OldColor)
      FOR T = 1 TO BlinkDuration: NEXT T
      CALL Plot3DPoint(1, YNow, ZNow, NewColor)
      FOR T = 1 TO BlinkDuration: NEXT T
      NEXT I%
                                          'the farthest distance of the fin facet
                                          'from the viewer is determined
    XFacetMax = -9999
    FOR Corner% = 1 TO 4
      CALL TransCoordBS(XFinEdgePos(FinNumber%, LeadEdgeLongSegNumber%,
                       Corner%), YFinEdgePos(FinNumber%,
                       LeadEdgeLongSegNumber%, Corner%),
                       ZFinEdgePos(LeadEdgeLongSegNumber%, Corner%), XSpace,
                       YSpace, ZSpace)
      IF (XSpace > XFacetMax) THEN XFacetMax = XSpace
      NEXT Corner%
                                          'further testing continues only if the
                                          'farthest distance of the fin facet from
                                          'the viewer is less than the distance to
                                          'the projectile facet under consideration
    IF (XFacetMax < XTest) THEN
```

```
                                        'the vertical extremes of the possible
                                        'blocking fin facet are determined
YFacetMin = 9999
YFacetMax = -9999
FOR Corner% = 1 TO 4
  CALL TransCoordBS(XFinEdgePos(FinNumber%, LeadEdgeLongSegNumber%,
              Corner%), YFinEdgePos(FinNumber%,
              LeadEdgeLongSegNumber%, Corner%),
              ZFinEdgePos(LeadEdgeLongSegNumber%, Corner%), XSpace,
              YSpace, ZSpace)
  IF (YSpace > YFacetMax) THEN YFacetMax = YSpace
  IF (YSpace < YFacetMin) THEN YFacetMin = YSpace
  NEXT Corner%
                                        'the midpoint vertical position of the
                                        'facet that is being tested for blockage
                                        'is compared to the vertical extremes of
                                        'the potentially blocking fin facet. if
                                        'the midpoint is contained within the
                                        'range of the fin facet's extremes, then
                                        'further testing will be performed that
                                        'considers horizontal overlap. if
                                        'not, then this fin facet can not be
                                        'blocking.
IF ((YFacetMin < YTest) AND (YFacetMax > YTest)) THEN
                                        'the horizontal extremes of the possible
                                        'blocking fin facet are determined
  ZFacetMin = 9999
  ZFacetMax = -9999
  FOR Corner% = 1 TO 4
    CALL TransCoordBS(XFinEdgePos(FinNumber%, LeadEdgeLongSegNumber%,
              Corner%), YFinEdgePos(FinNumber%,
              LeadEdgeLongSegNumber%, Corner%),
              ZFinEdgePos(LeadEdgeLongSegNumber%, Corner%), XSpace,
              YSpace, ZSpace)
    IF (ZSpace > ZFacetMax) THEN ZFacetMax = ZSpace
    IF (ZSpace < ZFacetMin) THEN ZFacetMin = ZSpace
    NEXT Corner%
                                        'the midpoint horizontal position of the
                                        'facet that is being tested for blockage
                                        'is compared to the horizontal extremes of
                                        'the potentially blocking fin facet. if
                                        'the midpoint is contained within the
                                        'range of the fin facet's extremes, then
                                        'it is concluded that the fin facet will
                                        'block the facet being tested.
  IF ((ZFacetMin < ZTest) AND (ZFacetMax > ZTest)) THEN
    BEEP
                                        'the fin facet midpoint position pixel
                                        'is returned to its original color
    CALL Plot3DPoint(1, YNow, ZNow, OldColor)
    Ans% = 0
    GOTO Hidden
                                        'if this fin facet is determined not to
                                        'block the facet under consideration,
                                        'then the fin facet's midpoint position
                                        'pixel is simply returned to its original
                                        'color
    ELSE
    CALL Plot3DPoint(1, YNow, ZNow, OldColor)
    END IF
                                        'if this fin facet is determined not to
                                        'block the facet under consideration,
                                        'then the fin facet's midpoint position
                                        'pixel is simply returned to its original
                                        'color
  ELSE
```

```
            CALL Plot3DPoint(1, YNow, ZNow, OldColor)
            END IF
                                        'if this fin facet is determined not to
                                        'block the facet under consideration,
                                        'then the fin facet's midpoint position
                                        'pixel is simply returned to its original
                                        'color
          ELSE
          CALL Plot3DPoint(1, YNow, ZNow, OldColor)
          END IF
        END IF
      NEXT LeadEdgeLongSegNumber%
SkipFin:
    NEXT FinNumber%
                                        'checking for blocking by aftbody facets
                                        'checking for simplification #9
  IF (Type$ = "AftBody") THEN GOTO SkipAftBody
                                        'checking for simplification #10
  IF ((FinUnitCOSValue > 0) AND (Type$ = "Fin")) THEN GOTO SkipAftBody
                                        'each transverse slice of the aftbody
                                        'is considered
  FOR ZSegment% = 0 TO NumLeadEdgeLongSeg% + NumNonLeadEdgeLongSeg% - 1
                                        'each aftbody area between adjacent fins
                                        'is considered
    FOR FinNumber% = 0 TO NumFins% - 1
                                        'the radial facets in each aftbody area
                                        'are considered
      FOR AftBodyRadSegNumber% = 0 TO NumAftBodyRadSegPerFin% - 1
        NormalDotProduct = XAftBodyNormal(FinNumber%, AftBodyRadSegNumber%) *
                         XViewNormal + YAftBodyNormal(FinNumber%,
                         AftBodyRadSegNumber%) * YViewNormal
                                        'only those aftbody facets with normals
                                        'that point "toward" the viewer are
                                        'considered
        IF (NormalDotProduct > 0) THEN
                                        'the midpoint position of the aftbody
                                        'facet is determined
          ZNow = 0
          YNow = 0
          FOR Corner% = 1 TO 4
            CALL TransCoordBS(XAftBodyPos(FinNumber%, AftBodyRadSegNumber%,
                         Corner%), YAftBodyPos(FinNumber%,
                         AftBodyRadSegNumber%, Corner%),
                         ZAftBodyPos(ZSegment%, Corner%), XSpace, YSpace,
                         ZSpace)
            ZNow = ZNow + ZSpace / 4
            YNow = YNow + YSpace / 4
            NEXT Corner%
                                        'the original color of the midpoint
                                        'position pixel is determined and saved
          OldColor = POINT(ZNow * Scale, YNow * Scale)
                                        'the midpoint position pixel of the
                                        'aftbody facet is flashed
          FOR I% = 1 TO NumBlinks%
            CALL Plot3DPoint(1, YNow, ZNow, OldColor)
            FOR T = 1 TO BlinkDuration: NEXT T
            CALL Plot3DPoint(1, YNow, ZNow, NewColor)
            FOR T = 1 TO BlinkDuration: NEXT T
            NEXT I%
                                        'the farthest distance of the aftbody
                                        'facet from the viewer is determined
          XFacetMax = -9999
          FOR Corner% = 1 TO 4
            CALL TransCoordBS(XAftBodyPos(FinNumber%, AftBodyRadSegNumber%,
                         Corner%), YAftBodyPos(FinNumber%,
                         AftBodyRadSegNumber%, Corner%),
```

```
                     ZAftBodyPos(ZSegment%, Corner%), XSpace, YSpace,
                     ZSpace)
     IF (XSpace > XFacetMax) THEN XFacetMax = XSpace
     NEXT Corner%
                                            'further testing continues only if the
                                            'farthest distance of the aftbody facet
                                            'from the viewer is less than the distance
                                            'to the projectile facet under consideration
  IF (XFacetMax < XTest) THEN
                                            'the vertical extremes of the possible
                                            'blocking aftbody facet are determined
     YFacetMin = 9999
     YFacetMax = -9999
     FOR Corner% = 1 TO 4
        CALL TransCoordBS(XAftBodyPos(FinNumber%, AftBodyRadSegNumber%,
                     Corner%), YAftBodyPos(FinNumber%,
                     AftBodyRadSegNumber%, Corner%),
                     ZAftBodyPos(ZSegment%, Corner%), XSpace, YSpace,
                     ZSpace)
        IF (YSpace > YFacetMax) THEN YFacetMax = YSpace
        IF (YSpace < YFacetMin) THEN YFacetMin = YSpace
        NEXT Corner%
                                            'the midpoint vertical position of the
                                            'facet that is being tested for blockage
                                            'is compared to the vertical extremes of
                                            'the potentially blocking aftbody facet.
                                            'if the midpoint is contained within the
                                            'range of the aftbody facet's extremes,
                                            'then further testing will be performed
                                            'that considers horizontal overlap. if
                                            'not, then this aftbody facet can not be
                                            'blocking.
     IF ((YFacetMin < YTest) AND (YFacetMax > YTest)) THEN
                                            'the horizontal extremes of the possible
                                            'blocking aftbody facet are determined
        ZFacetMin = 9999
        ZFacetMax = -9999
        FOR Corner% = 1 TO 4
           CALL TransCoordBS(XAftBodyPos(FinNumber%, AftBodyRadSegNumber%,
                     Corner%), YAftBodyPos(FinNumber%,
                     AftBodyRadSegNumber%, Corner%),
                     ZAftBodyPos(ZSegment%, Corner%), XSpace, YSpace,
                     ZSpace)
           IF (ZSpace > ZFacetMax) THEN ZFacetMax = ZSpace
           IF (ZSpace < ZFacetMin) THEN ZFacetMin = ZSpace
           NEXT Corner%
                                            'the midpoint horizontal position of the
                                            'facet that is being tested for blockage
                                            'is compared to the horizontal extremes of
                                            'the potentially blocking aftbody facet.
                                            'if the midpoint is contained within the
                                            'range of the aftbody facet's extremes,
                                            'then it is concluded that the aftbody
                                            'facet will block the facet being tested.
        IF ((ZFacetMin < ZTest) AND (ZFacetMax > ZTest)) THEN
           BEEP
                                            'the aftbody facet midpoint position pixel
                                            'is returned to its original color
           CALL Plot3DPoint(1, YNow, ZNow, OldColor)
           Ans% = 0
           GOTO Hidden
                                            'if this aftbody facet is determined not
                                            'to block the facet under consideration,
                                            'then the aftbody facet's midpoint
                                            'position pixel is simply returned to its
                                            'original color
```

```
                    ELSE
                    CALL Plot3DPoint(1, YNow, ZNow, OldColor)
                    END IF
                                                        'if this aftbody facet is determined not
                                                        'to block the facet under consideration,
                                                        'then the aftbody facet's midpoint
                                                        'position pixel is simply returned to its
                                                        'original color
                  ELSE
                  CALL Plot3DPoint(1, YNow, ZNow, OldColor)
                  END IF
                                                        'if this aftbody facet is determined not
                                                        'to block the facet under consideration,
                                                        'then the aftbody facet's midpoint
                                                        'position pixel is simply returned to its
                                                        'original color
                ELSE
                CALL Plot3DPoint(1, YNow, ZNow, OldColor)
                END IF
              END IF
            NEXT AftBodyRadSegNumber%
          NEXT FinNumber%
        NEXT ZSegment%

SkipAftBody:

Hidden:

END SUB

SUB Init3DDisplay
'**************************************************************************************************

'THIS SUBROUTINE INITIALIZES THE GRAPHIC VIEWPORT USING THE PREVIOUSLY
'DETERMINED SCALE FACTOR. THE GRAPHIC DISPLAY PARAMETERS ARE THEN OUTPUT
'IN LEADER FORMAT TO THE DATA STORAGE FILE. FINALLY, THE GRAPHIC DISPLAY
'COLORS ARE DEFINED.

'**************************************************************************************************

  SHARED YSpaceMin, YSpaceMax, ZSpaceMin, ZSpaceMax, Scale, Control$
  SHARED ScreenWidth, ScreenHeight, AspectRatio

  DIM C&(15)
                                                        'the graphic viewport is initializes to
                                                        'cover the entire monitor screen

  SCREEN 12
  VIEW (1, 1)-(ScreenWidth, ScreenHeight), 1
                                                        'the viewport coordinate system is
                                                        'defined. the overlap variables pertain to
                                                        'those portions of the less restrictive
                                                        'direction that are essentually left empty
  IF (Control$ = "Y") THEN
    ZOverlap = ScreenWidth - Scale * ABS(ZSpaceMax - ZSpaceMin)
    WindowX1 = ZSpaceMin * Scale - ZOverlap / 2
    WindowY1 = YSpaceMin * Scale
    WindowX2 = ZSpaceMax * Scale + ZOverlap / 2
    WindowY2 = YSpaceMax * Scale
    WINDOW ((WindowX1 + WindowX2) / 2 - AspectRatio * (WindowX2 - WindowX1) / 2,
                              WindowY1)-((WindowX1 + WindowX2) / 2 + AspectRatio *
                              (WindowX2 - WindowX1) / 2, WindowY2)
    END IF

  IF (Control$ = "Z") THEN
    YOverlap = ScreenHeight - Scale * ABS(YSpaceMax - YSpaceMin)
```

```
        WindowX1 = ZSpaceMin * Scale
        WindowY1 = YSpaceMin * Scale - YOverlap / 2
        WindowX2 = ZSpaceMax * Scale
        WindowY2 = YSpaceMax * Scale + YOverlap / 2
        WINDOW ((WindowX1 + WindowX2) / 2 - AspectRatio * (WindowX2 - WindowX1) / 2,
                                WindowY1)-((WindowX1 + WindowX2) / 2 + AspectRatio *
                                (WindowX2 - WindowX1) / 2, WindowY2)
        END IF
                                                        'graphic screen definition parameters are
                                                        'saved in the data storage file
    PRINT #1, "graphic scale factor: "
    PRINT #1, Scale
    PRINT #1, "WindowX1: "
    PRINT #1, WindowX1
    PRINT #1, "WindowY1: "
    PRINT #1, WindowY1
    PRINT #1, "WindowX2: "
    PRINT #1, WindowY2

    Blue = 63
    Breen = 63
    Red = 63
                                                        'palette colors 3 through 15 are defined
                                                        'to be shades of gray with 3 being black
                                                        'and 15 being white.
    FOR I% = 3 TO 15
        C&(I%) = 65536 * INT(((I% - 3) / 12) * Blue) + 256 * INT(((I% - 3) / 12) * Green) +
                                INT(((I% - 3) / 12) * Red)
        NEXT I%
                                                        'palette color 0 is defined to be black
                                                        'for "erasing" purposes, color 1 will
                                                        'serve as the backgroung color, and color
                                                        '2 will be used for drawing construction
                                                        'lines
    C&(0) = 0
    C&(1) = 32
    C&(2) = 26
    PALETTE USING C&(0)

END SUB

SUB InputParameters

'**************************************************************************************************

'THIS SUBROUTINE REQUESTS THE USER TO INPUT PARAMETER VALUES FOR THE
'PROJECTILE'S DIMENSIONS, TEMPERATURE, MATERIAL CHARACTERISTICS, AND
'ORIENTATION. THIS INFORMATION IS THEN SAVED AS LEADER INFORMATION IN
'A DATA FILE USING A FILE NAME WHICH IS SUPPLIED BY THE USER.

'**************************************************************************************************

    SHARED LengthNoseCone, DeltaZNoseCone, NumNoseConeLongSeg%, RadNoseCone
    SHARED NumNoseConeRadSeg%, LengthBody, DeltaZBody, NumBodyLongSeg%
    SHARED RadBody, NumBodyRadSeg%, NumFins%, ThickFin, LengthBaseFin
    SHARED LengthLeadEdgeFin, DeltaZFin, NumLeadEdgeLongSeg%
    SHARED NumNonLeadEdgeLongSeg%, HeightFin, DeltaRadFin, NumFinRadSeg%
    SHARED NumAftBodyRadSegPerFin%, Euler(), File1$, NoseForwTemp
    SHARED NoseRearTemp, NoseEmis, BodyForwTemp, BodyRearTemp, BodyEmis
    SHARED FinOuterWRTInnerTemp, FinLeadTemp, FinTrailTemp, FinEmis
    SHARED AftBodyForwTemp, AftBodyRearTemp, AftBodyEmis

    CLS
                                                        'the user is requested to enter projectile
                                                        'parameter information
    INPUT "Enter the length of the nose cone in millimeters: "; LengthNoseCone
```

```
LengthNoseCone = LengthNoseCone / 1000
PRINT

INPUT "Enter the nose cone longitudinal segment length in millimeters: "; DeltaZNoseCone
DeltaZNoseCone = DeltaZNoseCone / 1000
PRINT

NumNoseConeLongSeg% = CINT(LengthNoseCone / DeltaZNoseCone)

INPUT "Enter the radius of the nose cone base in millimeters: "; RadNoseCone
RadNoseCone = RadNoseCone / 1000
PRINT

INPUT "Enter the number of nose cone cord segments: "; NumNoseConeRadSeg%
PRINT

INPUT "Enter the length of the body in millimeters: "; LengthBody
LengthBody = LengthBody / 1000
PRINT

INPUT "Enter the body longitudinal segment length in millimeters: "; DeltaZBody
DeltaZBody = DeltaZBody / 1000
PRINT

NumBodyLongSeg% = CINT(LengthBody / DeltaZBody)

INPUT "Enter the radius of the body in millimeters: "; RadBody
RadBody = RadBody / 1000
PRINT

INPUT "Enter the number of body cord segments: "; NumBodyRadSeg%
PRINT

INPUT "Enter the number of fins: "; NumFins%
PRINT

INPUT "Enter the fin thickness in millimeters: "; ThickFin
ThickFin = ThickFin / 1000
PRINT

INPUT "Enter the base length of the fin in millimeters: "; LengthBaseFin
LengthBaseFin = LengthBaseFin / 1000
PRINT

INPUT "Enter the base length of the fin leading edge in millimeters: "; LengthLeadEdgeFin
LengthLeadEdgeFin = LengthLeadEdgeFin / 1000
PRINT

INPUT "Enter the fin longitudinal segment length in millimeters: "; DeltaZFin
DeltaZFin = DeltaZFin / 1000
PRINT

NumLeadEdgeLongSeg% = CINT(LengthLeadEdgeFin / DeltaZFin)
NumNonLeadEdgeLongSeg% = CINT((LengthBaseFin - LengthLeadEdgeFin) / DeltaZFin)

INPUT "Enter the fin height in millimeters: "; HeightFin
HeightFin = HeightFin / 1000
PRINT

INPUT "Enter the fin transverse segment length in millimeters: "; DeltaRadFin
DeltaRadFin = DeltaRadFin / 1000
PRINT

NumFinRadSeg% = CINT(HeightFin / DeltaRadFin)

INPUT "Enter the number of aft body cord segments between fins: ";
```

```
                                 NumAftBodyRadSegPerFin%
PRINT

INPUT "Enter the three Euler angles in degrees: "; Euler(1), Euler(2), Euler(3)
FOR I% = 1 TO 3
   Euler(I%) = Euler(I%) * 3.14159 / 180
   NEXT I%
PRINT

INPUT "Enter the nose cone forward tip temperature: "; NoseForwTemp
PRINT

INPUT "Enter the nose cone rear base temperature: "; NoseRearTemp
PRINT

INPUT "Enter the nose cone emissivity: "; NoseEmis
PRINT

INPUT "Enter the body forward temperature: "; BodyForwTemp
PRINT

INPUT "Enter the body rear temperature: "; BodyRearTemp
PRINT

INPUT "Enter the body emissivity: "; BodyEmis
PRINT

PRINT "Enter the fin outer edge temperature relative to"
INPUT "  the inner edge temperature: "; FinOuterWRTInnerTemp
PRINT

PRINT "Enter the fin leading edge temperature"
INPUT "  at the inner edge: "; FinLeadTemp
PRINT

PRINT "Enter the fin trailing edge temperature "
INPUT "  at the inner edge: "; FinTrailTemp
PRINT

INPUT "Enter the fin emissivity: "; FinEmis
PRINT

INPUT "Enter the aftbody forward temperature: "; AftBodyForwTemp
PRINT

INPUT "Enter the aftbody rear temperature: "; AftBodyRearTemp
PRINT

INPUT "Enter the aftbody emissivity: "; AftBodyEmis
PRINT

INPUT "Enter the storage data file name: "; File1$
PRINT
                                          'the projectile parameters are saved as
                                          'leader information in a data file
OPEN "C:\QB45\IRMODEL\" + File1$ FOR OUTPUT AS #1

PRINT #1, "length of nose cone in meters: "
PRINT #1, LengthNoseCone
PRINT #1, "nose cone longitudinal segment length in meters: "
PRINT #1, DeltaZNoseCone
PRINT #1, "number of nose cone longitudinal segments: "
PRINT #1, NumNoseConeLongSeg%
PRINT #1, "nose cone radius in meters: "
PRINT #1, RadNoseCone
PRINT #1, "number of nose cone cord segments: "
```

```
PRINT #1, NumNoseConeRadSeg%
PRINT #1, "length of body in meters: "
PRINT #1, LengthBody
PRINT #1, "body longitudinal segment length in meters: "
PRINT #1, DeltaZBody
PRINT #1, "number of longitudinal body segments: "
PRINT #1, NumBodyLongSeg%
PRINT #1, "body radius in meters: "
PRINT #1, RadBody
PRINT #1, "number of body cord segments: "
PRINT #1, NumBodyRadSeg%
PRINT #1, "number of fins: "
PRINT #1, NumFins%
PRINT #1, "fin thickness in meters: "
PRINT #1, ThickFin
PRINT #1, "total fin base length in meters: "
PRINT #1, LengthBaseFin
PRINT #1, "base length of fin leading edge in meters: "
PRINT #1, LengthLeadEdgeFin
PRINT #1, "fin longitudinal segment length in meters: "
PRINT #1, DeltaZFin
PRINT #1, "number of fin leading edge longitudinal segments: "
PRINT #1, NumLeadEdgeLongSeg%
PRINT #1, "number of fin nonleading edge longitudinal segments: "
PRINT #1, NumNonLeadEdgeLongSeg%
PRINT #1, "fin height in meters: "
PRINT #1, HeightFin
PRINT #1, "fin transverse segment length in meters: "
PRINT #1, DeltaRadFin
PRINT #1, "number of fin radial segments: "
PRINT #1, NumFinRadSeg%
PRINT #1, "number of aft body cord segments between fins: "
PRINT #1, NumAftBodyRadSegPerFin%
PRINT #1, "Euler angle 1: "
PRINT #1, Euler(1) * 180 / 3.14159
PRINT #1, "Euler angle 2: "
PRINT #1, Euler(2) * 180 / 3.14159
PRINT #1, "Euler angle 3: "
PRINT #1, Euler(3) * 180 / 3.14159
PRINT #1, "nose cone forward tip temperature: "
PRINT #1, NoseForwTemp
PRINT #1, "nose cone rear base temperature: "
PRINT #1, NoseRearTemp
PRINT #1, "nose cone emissivity: "
PRINT #1, NoseEmis
PRINT #1, "body forward temperature: "
PRINT #1, BodyForwTemp
PRINT #1, "body rear temperature: "
PRINT #1, BodyRearTemp
PRINT #1, "body emissivity: "
PRINT #1, BodyEmis
PRINT #1, "fin outer edge temperature wrt inner edge temperature: "
PRINT #1, FinOuterWRTInnerTemp
PRINT #1, "fin leading edge temperature at the inner edge: "
PRINT #1, FinLeadTemp
PRINT #1, "fin trailing edge temperature at the inner edge: "
PRINT #1, FinTrailTemp
PRINT #1, "fin emissivity: "
PRINT #1, FinEmis
PRINT #1, "aftbody forward temperature: "
PRINT #1, AftBodyForwTemp
PRINT #1, "aftbody rear temperature: "
PRINT #1, AftBodyRearTemp
PRINT #1, "aftbody emissivity: "
PRINT #1, AftBodyEmis
PRINT #1, "name of this data file: "
```

```
    PRINT #1, File1$
END SUB
SUB LoadDataFile (Type$, X(), Y(), Z(), NormalDotProduct, Area, Temp, Emis)
'*************************************************************************************************
'THIS SUBROUTINE IS USED TO LOAD THE INFORMATION ABOUT A FACET INTO THE DATA
'STORAGE FILE
'*************************************************************************************************
    PRINT #1, USING "\ \_, +#.#### +#.#### +#.#### +#.#### +#.#### +#.#### +#.####
                  +#.#### +#.#### +#.#### +#.#### +#.#### #.#### .######## ####
                  #.##"; Type$; X(1); Y(1); Z(1); X(2); Y(2); Z(2); X(3); Y(3); Z(3); X(4); Y(4);
                  Z(4); NormalDotProduct; Area; Temp; Emis
END SUB
SUB Plot3DLine (X, Y, Z, C1)
'*************************************************************************************************
'THIS SUBROUTINE IS USED TO PLOT A LINE FROM THE PREVIOUS GRAPHIC SCREEN
'POSITION TO THE NEWLY PROVIDED COORDINATES
'*************************************************************************************************
    SHARED Scale
    LINE -(Z * Scale, Y * Scale), C1
END SUB
SUB Plot3DPoint (X, Y, Z, C1)
'*************************************************************************************************
'THIS SUBROUTINE IS USED TO PLOT A POINT ON THE GRAPHIC DISPLAY
'*************************************************************************************************
    SHARED Scale
    PSET (Z * Scale, Y * Scale), C1
END SUB
SUB TransCoordBS (XOld, YOld, ZOld, XSpace, YSpace, ZSpace)
'*************************************************************************************************
'THIS SUBROUTINE TRANSFORMS FROM PROJECTILE BODY COORDINATES TO SPACE
'COORDINATES.
'*************************************************************************************************
    SHARED E()
    DIM COld(3), CNew(3)
    COld(1) = XOld
    COld(2) = YOld
    COld(3) = ZOld
```

```
    FOR I% = 1 TO 3
       CNew(I%) = 0
       FOR J% = 1 TO 3
          CNew(I%) = CNew(I%) + COld(J%) * E(I%, J%)
          NEXT J%
       NEXT I%

    XSpace = CNew(1)
    YSpace = CNew(2)
    ZSpace = CNew(3)

END SUB

SUB TransCoordSB (XSpace, YSpace, ZSpace, XBody, YBody, ZBody)

'*****************************************************************************************************

'THIS SUBROUTINE TRANSFORMS FROM SPACE COORDINATES TO PROJECTILE BODY
'COORDINATES.

'*****************************************************************************************************

    SHARED E()

    DIM COld(3), CNew(3)

    COld(1) = XSpace
    COld(2) = YSpace
    COld(3) = ZSpace

    FOR I% = 1 TO 3
       CNew(I%) = 0
       FOR J% = 1 TO 3
          CNew(I%) = CNew(I%) + COld(J%) * E(J%, I%)
          NEXT J%
       NEXT I%

    XBody = CNew(1)
    YBody = CNew(2)
    ZBody = CNew(3)

END SUB
```

APPENDIX B:

PENETRATOR IR EMISSION CALCULATION SOFTWARE

INTENTIONALLY LEFT BLANK.

This software calculates the spectral and spatial distributions of infrared (IR) radiation emitted by a kinetic energy (KE) penetrator. A data file must be input to this program that has been generated by the previously discussed and listed facet model generation software. MicroSoft QuickBasic 4.5 is used as the programming environment. If you have any questions about this code, please contact Tom Kottke at:

AMSRL-WT-WD
Survivability Concepts Branch
Weapons Concepts Division, Bldg. 120
Weapons Technology Directorate
U.S. Army Research Laboratory
Aberdeen Proving Ground, MD 21005-5066
(410) 278-2557


```
                                                 'subroutines are declared
DECLARE SUB ReadInputData1 ()
DECLARE SUB ReadInputData2 ()
DECLARE SUB DeterPlotColor (Temp, C)
DECLARE SUB Init3DDisp ()
DECLARE SUB Plot3DPoint (X, Y, Z, C1)
DECLARE SUB Plot3DLine (X, Y, Z, C1)
DECLARE SUB DeterPFacet (FacNumber%, SpecWinNumber%, PofL)
DECLARE SUB OutputData ()

CLS
                                                 'Euler angle variable is dimensioned
DIM Euler(3)
                                                 'leader data is input from datafile
                                                 'output file name is specified
CALL ReadInputData1

PRINT "Enter the file name that the calculated spectral data is to be"
INPUT " saved in: "; File2$
                                                 'other array variables are dimensioned
DIM Type$(NumFacets%), X(NumFacets%, 4), Y(NumFacets%, 4), Z(NumFacets%, 4)
DIM NormalCos(NumFacets%), Area(NumFacets%), Temp(NumFacets%)
DIM Emis(NumFacets%), ZAve(NumFacets%)
                                                 'facet data is input from datafile
CALL ReadInputData2
                                                 'minimum and maximum facet temperatures
                                                 'are determined
TempMin = 9999
TempMax = -9999
FOR I% = 1 TO NumFacets%
   IF (Temp(I%) < TempMin) THEN TempMin = Temp(I%)
   IF (Temp(I%) > TempMax) THEN TempMax = Temp(I%)
   NEXT I%
                                                 'the average horizontal position is
                                                 'calculated for each facet
FOR I% = 1 TO NumFacets%
   ZAve(I%) = 0
   FOR J% = 1 TO 4
      ZAve(I%) = ZAve(I%) + Z(I%, J%) / 4
      NEXT J%
   NEXT I%
                                                 'the minimum and maximum average
```

```
                                                    'horizontal facet positions are
                                                    'determined
ZAveMin = 1E+10
ZAveMax = -1E-10

FOR I% = 1 TO NumFacets%
   IF (ZAve(I%) < ZAveMin) THEN ZAveMin = ZAve(I%)
   IF (ZAve(I%) > ZAveMax) THEN ZAveMax = ZAve(I%)
   NEXT I%
                                                    'the minimum and maximum average
                                                    'horizontal facet positions are output to
                                                    'the screen to allow the user to determine
                                                    'the horizontal range over which IR
                                                    'spectral calculations are to be
                                                    'performed
PRINT "Minimum Z Average Value: "; ZAveMin
PRINT
PRINT "Maximum Z Average Value: "; ZAveMax
PRINT
                                                    'the user enters information regarding the
                                                    'spatial and spectral extents over which
                                                    'the IR calculations are to be performed
INPUT "Enter the number of spatially resolved regions: "; NumZRegions%
PRINT
PRINT "Enter the minimum and maximum Z values of the"
INPUT "  total spatial region: "; ZSPatialMin, ZSpatialMax
ZSpatialSize = (ZSpatialMax - ZSPatialMin) / NumZRegions%
PRINT
INPUT "Enter the minimum spectral wavelength in microns: "; SpecMin
PRINT
INPUT "Enter the maximum spectral wavelength in microns: "; SpecMax
PRINT
INPUT "Enter the span of each spectral window in microns: "; SpecWindowSize
NumSpecWindows% = (SpecMax - SpecMin) / SpecWindowSize
PRINT
                                                    'user enters projectile's range from the
                                                    'assumed IR detector. This information is
                                                    'used to calculate the intensity of the
                                                    'IR radiation that is incident on the
                                                    'detector
INPUT "Enter the range to the projectile in meters: "; ProjRange
PRINT
                                                    'radiant flux array variables are
                                                    'dimensioned
DIM P(NumZRegions%, NumSpecWindows%), PRegion(NumZRegions%)
                                                    '3D graphic display is initialized
CALL Init3DDisp
                                                    'wire mesh model of penetrator is
                                                    'presented graphically in a neutral
                                                    'color to illustrate what portions of
                                                    'the penetrator are contained in each
                                                    'spatial zone
FOR I% = 1 TO NumFacets%
   CALL Plot3DPoint(X(I%, 4), Y(I%, 4), Z(I%, 4), 15)
   FOR J% = 1 TO 4
      CALL Plot3DLine(X(I%, J%), Y(I%, J%), Z(I%, J%), 15)
      NEXT J%
   NEXT I%
                                                    'the radiant flux from each spatial
                                                    'region is calculated
FOR ZRegNumber% = 1 TO NumZRegions%
                                                    'the limits of the spatial region under
                                                    'consideration are calculated
   ZRegMin = ZSPatialMin + (ZRegNumber% - 1) * ZSpatialSize
   ZRegMax = ZSPatialMin + ZRegNumber% * ZSpatialSize
   FOR FacNumber% = 1 TO NumFacets%
```

```
                                                   'each facet is checked to determine if it
                                                   'is located in the spatial region under
                                                   'consideration
      IF ((ZAve(FacNumber%) >= ZRegMin) AND (ZAve(FacNumber%) < ZRegMax)) THEN
                                                   'if a given facet is determined to be
                                                   'located in the spatial region under
                                                   'consideration its facet is redrawn in a
                                                   'color that is indicative of its
                                                   'temperature
        CALL DeterPlotColor(Temp(FacNumber%), C)
        CALL Plot3DPoint(X(FacNumber%, 4), Y(FacNumber%, 4), Z(FacNumber%, 4), C)
        FOR Corner% = 1 TO 4
          CALL Plot3DLine(X(FacNumber%, Corner%), Y(FacNumber%, Corner%),
                                    Z(FacNumber%, Corner%), C)
          NEXT Corner%
                                                   'the contribution of the facet to the
                                                   'radiance within each spectral zone is
                                                   'calculated and added to the total
                                                   'radiance produced by the spatial region
                                                   'under consideration
        FOR SpecWinNumber% = 1 TO NumSpecWindows%
          CALL DeterPFacet(FacNumber%, SpecWinNumber%, PofL)
          P(ZRegNumber%, SpecWinNumber%) = P(ZRegNumber%, SpecWinNumber%) + PofL
          NEXT SpecWinNumber%
        END IF
      NEXT FacNumber%
    NEXT ZRegNumber%
                                                   'the maximum spectral radiance within any
                                                   'spectral or spatial region is determined
                                                   'for display scaling
  PMax = -99999
  FOR ZRegNumber% = 1 TO NumZRegions%
    FOR SpecWinNumber% = 1 TO NumSpecWindows%
      IF (P(ZRegNumber%, SpecWinNumber%) > PMax) THEN PMax = P(ZRegNumber%,
                                    SpecWinNumber%)
      NEXT SpecWinNumber%
    NEXT ZRegNumber%
                                                   'the spectral radiance is displayed for
                                                   'each spatial region
  FOR ZRegNumber% = 1 TO NumZRegions%
    PSET ((ZSPatialMin + (ZRegNumber% - 1) * ZSpatialSize + (1 / NumSpecWindows%) *
                                    ZSpatialSize) * Scale, WindowY1 + (P(ZRegNumber%, 1) /
                                    PMax) * (WindowY2 - WindowY1) / 4), 15
    FOR SpecWinNumber% = 2 TO NumSpecWindows%
      LINE -((ZSPatialMin + (ZRegNumber% - 1) * ZSpatialSize + (SpecWinNumber% /
                                    NumSpecWindows%) * ZSpatialSize) * Scale, WindowY1 +
                                    (P(ZRegNumber%, SpecWinNumber%) / PMax) * (WindowY2 -
                                    WindowY1) / 4), 15
      NEXT SpecWinNumber%
    NEXT ZRegNumber%
                                                   'wait for operator's response
  LOCATE 28, 1
  PRINT "Press any key to continue..."
  DO
    LOOP WHILE INKEY$ = ""
  LOCATE 28, 1
  PRINT "                              "
                                                   'the radiant flux for the total spectral
                                                   'region under consideration is determined
                                                   'for each spatial region and the entire
                                                   'penetrator
  PTotal = 0
  FOR ZRegNumber% = 1 TO NumZRegions%
    PRegion(ZRegNumber%) = 0
    FOR SpecWinNumber% = 1 TO NumSpecWindows%
      PTotal = PTotal + P(ZRegNumber%, SpecWinNumber%)
```

```
      PRegion(ZRegNumber%) = PRegion(ZRegNumber%) + P(ZRegNumber%, SpecWinNumber%)
      NEXT SpecWinNumber%
   NEXT ZRegNumber%
                                                'radiant flux integrated over the spatial
                                                'regions is displayed for integrations
                                                'beginning at the left and the right of
                                                'the penetrator
PAccum = 0
FOR ZRegNumber% = 1 TO NumZRegions%
   PAccum = PAccum + PRegion(ZRegNumber%)
   CIRCLE ((ZSPatialMin + (ZRegNumber% - .5) * ZSpatialSize) * Scale, (PAccum / PTotal) *
                 (WindowY2 - WindowY1) + WindowY1), (WindowY2 - WindowY1) / 150, 15
   PAINT ((ZSPatialMin + (ZRegNumber% - .5) * ZSpatialSize) * Scale, (PAccum / PTotal) *
                 (WindowY2 - WindowY1) + WindowY1), 1, 15
   CIRCLE ((ZSPatialMin + (ZRegNumber% - .5) * ZSpatialSize) * Scale, ((PTotal - PAccum) /
                 PTotal) * (WindowY2 - WindowY1) + WindowY1), (WindowY2 - WindowY1) /
                 150, 15
   PAINT ((ZSPatialMin + (ZRegNumber% - .5) * ZSpatialSize) * Scale, ((PTotal - PAccum) /
                 PTotal) * (WindowY2 - WindowY1) + WindowY1), 14, 15
   NEXT ZRegNumber%
                                                'the calculated radiant flux values are
                                                'output to a data file
CALL OutputData
                                                'wait for operator's response
DO
   LOOP WHILE INKEY$ = ""

SUB DeterPFacet (FacNumber%, SpecWinNumber%, PofL)
'*************************************************************************************************

'THIS SUBROUTINE CALCULATES THE SPECTRAL RADIANT FLUX EMITTED BY A SINGLE
'FACET AND COLLECTED BY THE DETECTOR

'*************************************************************************************************

      'This subroutine calculates the spectral radiant flux (P of Lambda)
      'from a single facet to the detector. P is the rate at which radiant
      'energy is transferred from the facet to the detector. This quantity
      'is determined by first calculating the spectral radiant emittance
      '(W of Lambda) which is the radiant flux emitted per unit source area
      'per unit wavelength interval at a particular wavelength and is
      'calculated for a particular wavelength using the expression
      '
      '                         C1      1
      '             W(L)=n*-----*------------------
      '                         L^5  exp(C2/LT)-1
      '
      'where: W(L) is the spectral radiant emittance in units of
      '            watts/(m^2*micron)
      '       n    is the emissivity
      '       L    represents lambda which is the wavelength in microns
      '       T    is the absolute temperature in degrees Kelvin; K=C+273.16
      '       C1   is the first radiation constant equal to
      '            3.7415E+8 Watt*micron^4/m^2
      '       C2   is the second radiation constant equal to
      '            1.43879E+4 micron*degree K
      '
      'Making the assumption that the facet is a perfectly diffuse, or
      'Lambertian, source, the spectral radiance (N of Lambda), which is
      'the radiant flux per unit solid angle per unit area of source per
      'unit wavelength interval at a particular wavelength (phew!), is
      'calculated from the spectral radiant emittance through the simple
      'relationship (N of Lambda)=(W of Lambda)/(Pi). Finally, the spectral
      'radiant flux is calculated by multiplying the spectral radiant
      'emittance by the effective source area, the solid angle subtended by
```

```
    'the detector, and the span of the wavelength interval under
    'consideration. The detector is assumed to have an effective area of
    '1 square centimeter.

  SHARED Area(), Temp(), Emis(), NormalCos(), SpecMin, SpecWindowSize
  SHARED ProjRange, X(), Y(), Z()
                                          'define the values of the radiation
                                          'constants
  C1 = 3.7415E+08 ' W*u^4/m^2
  C2 = 14387.9    ' u*K
                                          'calculate the value of the wavelength
                                          'under consideration
  Lambda = SpecMin + (SpecWinNumber% - 1) * SpecWindowSize
                                          'calculate the spectral radiant emittance
  FirstTerm = C1 / (Lambda ^ 5)
  Exponent = C2 / (Lambda * (273.16 + Temp(FacNumber%)))
  IF (Exponent < 60) THEN
    SecondTerm = 1 / (EXP(Exponent) - 1)
    ELSE
    SecondTerm = 0
    END IF
  WofL = Emis(FacNumber%) * FirstTerm * SecondTerm
                                          'calculate the spectral radiance
  NofL = WofL / 3.14159
                                          'calculate the effective source area
  EffArea = Area(FacNumber%) * NormalCos(FacNumber%)
                                          'calculate the average distance of the
                                          'facet from the detector
  Distance = 0
  FOR Corner% = 1 TO 4
    Distance = Distance + SQR((X(FacNumber%, Corner%) + ProjRange) ^ 2 +
                   Y(FacNumber%, Corner%) ^ 2 + Z(FacNumber%, Corner%) ^ 2) / 4
    NEXT Corner%
                                          'calculate the solid angle subtended by a
                                          '1 square centimeter detector
  SolidAngle = .0001 / Distance ^ 2
                                          'calculate the spectral radiant flux
                                          'incident on the 1 square centimeter
                                          'detector
  PofL = NofL * EffArea * SolidAngle * SpecWindowSize

END SUB

SUB DeterPlotColor (Temp, C)
'*************************************************************************************

'THIS SUBROUTINE ASSIGNS A GRAPHIC DISPLAY COLOR BASED ON THE TEMPERATURE
'OF A FACET

'*************************************************************************************

  SHARED TempMin, TempMax

  C = CINT(((Temp - TempMin) / (TempMax - TempMin)) * 13 + 1)

END SUB

SUB Init3DDisp
'*************************************************************************************

'THIS SUBROUTINE INITIALIZES AND SCALES THE GRAPHIC DISPLAY WINDOW, DEFINES
'THE GRAPHIC PALLETE, AND VISUALLY DISPLAYS THE SPATIAL EXTENT OF THE
'INDIVIDUAL IR CALCULATION REGIONS
```

```
'*****************************************************************************************************************

    SHARED WindowX1, WindowY1, WindowX2, WindowY2, NumZRegions%, ZSPatialMin
    SHARED ZSpatialMax, ZSpatialSize, Scale

    DIM C&(15)
                                                          'graphic display is initialized and sized
    SCREEN 12
    VIEW (1, 1)-(638, 398)
    WINDOW (WindowX1, WindowY1)-(WindowX2, WindowY2)
                                                          'graphic pallete is defined
    C&(0) = 65536 * 0 + 256 * 0 + 0
    C&(14) = 65536 * 0 + 256 * 0 + 63
    C&(13) = 65536 * 0 + 256 * 19 + 63
    C&(12) = 65536 * 0 + 256 * 30 + 63
    C&(11) = 65536 * 0 + 256 * 40 + 63
    C&(10) = 65536 * 0 + 256 * 50 + 63
    C&(9) = 65536 * 0 + 256 * 60 + 63
    C&(8) = 65536 * 0 + 256 * 63 + 42
    C&(7) = 65536 * 0 + 256 * 62 + 21
    C&(6) = 65536 * 0 + 256 * 55 + 0
    C&(5) = 65536 * 12 + 256 * 43 + 0
    C&(4) = 65536 * 23 + 256 * 36 + 0
    C&(3) = 65536 * 37 + 256 * 28 + 0
    C&(2) = 65536 * 49 + 256 * 21 + 0
    C&(1) = 65536 * 63 + 256 * 0 + 0
    C&(15) = 65536 * 63 + 256 * 63 + 63

    PALETTE USING C&(0)

                                                          'spatial extent of IR calculation regions
                                                          'is displayed graphically
    FOR I% = 1 TO NumZRegions%
       C% = 15
       LINE ((ZSPatialMin + (I% - 1) * ZSpatialSize) * Scale, WindowY1)-((ZSPatialMin + I% *
                                         ZSpatialSize) * Scale, WindowY2), C%, B
       NEXT I%

END SUB

SUB OutputData

'*****************************************************************************************************************

'THIS SUBROUTINE OUTPUTS THE TOTAL RADIANT FLUX, THE RADIANT FLUX FROM EACH
'SPATIAL REGION AND THE SPECTRAL RADIANT FLUX FROM EACH REGION TO A DATAFILE

'*****************************************************************************************************************

    SHARED File2$, File1$, PTotal, NumZRegions%, PRegion(), NumSpecWindows%
    SHARED P()

    OPEN File2$ FOR OUTPUT AS #2

    PRINT #2, "Input data file:"
    PRINT #2, File1$
    PRINT #2, " "
    PRINT #2, "Total radiant flux:"
    PRINT #2, PTotal
    PRINT #2, " "
    FOR I% = 1 TO NumZRegions%
       PRINT #2, "    Spatial region #"; I%
       PRINT #2, "    Region radiant flux"
       PRINT #2, "    "; PRegion(I%)
       FOR J% = 1 TO NumSpecWindows%
          PRINT #2, "          Spectral window #"; J%
```

```
        PRINT #2, "        "; P(I%, J%)
        NEXT J%
    NEXT I%

  CLOSE #2

END SUB

SUB Plot3DLine (X, Y, Z, C1)
'***************************************************************************************************************

'THIS SUBROUTINE PLOTS A DESIGNATED LINE ON THE GRAPHIC SCREEN IN A USER
'SELECTED COLOR
'***************************************************************************************************************

                                                    'Note that the horizontal direction
                                                    'corresponds to the Z axis and the
                                                    'vertical direction corresponds to
                                                    'the Y axis.
  SHARED Scale

  LINE -(Z * Scale, Y * Scale), C1

END SUB

SUB Plot3DPoint (X, Y, Z, C1)
'***************************************************************************************************************

'THIS SUBROUTINE PLOTS A DESIGNATED POINT ON THE GRAPHIC SCREEN IN A USER
'SELECTED COLOR
'***************************************************************************************************************

                                                    'Note that the horizontal direction
                                                    'corresponds to the Z axis and the
                                                    'vertical direction corresponds to
                                                    'the Y axis.
  SHARED Scale

  PSET (Z * Scale, Y * Scale), C1

END SUB

SUB ReadInputData1
'***************************************************************************************************************

'THIS SUBROUTINE MAKES THE FIRST PASS THROUGH A PREVIOUSLY GENERATED DATA
'FILE THAT CONTAINS INFORMATION ABOUT A PENETRATOR'S GEOMETRY AND THERMAL
'PROPERTIES. DURING THIS FIRST PASS THA LEADER PARAMETERS ARE INPUT AND THE
'NUMBER OF SUBSEQUENT LINES OF FACET DATA IS DETERMINED
'***************************************************************************************************************

  SHARED LengthNoseCone, DeltaZNoseCone, NumNoseConeLongSeg%, RadNoseCone
  SHARED NumNoseConeRadSeg%, LengthBody, DeltaZBody, NumBodyLongSeg%
  SHARED RadBody, NumBodyRadSeg%, NumFins%, ThickFin, LengthBaseFin
  SHARED LengthLeadEdgeFin, DeltaZFin, NumLeadEdgeLongSeg%
  SHARED NumNonLeadEdgeLongSeg%, HeightFin, DeltaRadFin, NumFinRadSeg%
  SHARED NumAftBodyRadSegPerFin%, Euler(), File1$, Scale, WindowX1
  SHARED WindowY1, WindowX2, WindowY2, NumFacets%

                                                    'previously generated datafile is opened
```

```
INPUT "Enter input data file name: "; File1$

OPEN File1$ FOR INPUT AS #1
                                              'leader parameters that were used to
                                              'generate the datafile are input and
                                              'displayed on the screen

INPUT #1, leader$
INPUT #1, LengthNoseCone
PRINT leader$; LengthNoseCone

INPUT #1, leader$
INPUT #1, DeltaZNoseCone
PRINT leader$; DeltaZNoseCone

INPUT #1, leader$
INPUT #1, NumNoseConeLongSeg%
PRINT leader$; NumNoseConeLongSeg%

INPUT #1, leader$
INPUT #1, RadNoseCone
PRINT leader$; RadNoseCone

INPUT #1, leader$
INPUT #1, NumNoseConeRadSeg%
PRINT leader$; NumNoseConeRadSeg%

INPUT #1, leader$
INPUT #1, LengthBody
PRINT leader$; LengthBody

INPUT #1, leader$
INPUT #1, DeltaZBody
PRINT leader$; DeltaZBody

INPUT #1, leader$
INPUT #1, NumBodyLongSeg%
PRINT leader$; NumBodyLongSeg%

INPUT #1, leader$
INPUT #1, RadBody
PRINT leader$; RadBody

INPUT #1, leader$
INPUT #1, NumBodyRadSeg%
PRINT leader$; NumBodyRadSeg%

INPUT #1, leader$
INPUT #1, NumFins%
PRINT leader$; NumFins%

INPUT #1, leader$
INPUT #1, ThickFin
PRINT leader$; ThickFin

INPUT #1, leader$
INPUT #1, LengthBaseFin
PRINT leader$; LengthBaseFin

INPUT #1, leader$
INPUT #1, LengthLeadEdgeFin
PRINT leader$; LengthLeadEdgeFin

INPUT #1, leader$
INPUT #1, DeltaZFin
PRINT leader$; DeltaZFin
```

```
INPUT #1, leader$
INPUT #1, NumLeadEdgeLongSeg%
PRINT leader$; NumLeadEdgeLongSeg%

INPUT #1, leader$
INPUT #1, NumNonLeadEdgeLongSeg%
PRINT leader$; NumNonLeadEdgeLongSeg%

INPUT #1, leader$
INPUT #1, HeightFin
PRINT leader$; HeightFin

INPUT #1, leader$
INPUT #1, DeltaRadFin
PRINT leader$; DeltaRadFin

INPUT #1, leader$
INPUT #1, NumFinRadSeg%
PRINT leader$; NumFinRadSeg%

INPUT #1, leader$
INPUT #1, NumAftBodyRadSegPerFin%
PRINT leader$; NumAftBodyRadSegPerFin%

INPUT #1, leader$
INPUT #1, Euler(1)
PRINT leader$; Euler(1)

INPUT #1, leader$
INPUT #1, Euler(2)
PRINT leader$; Euler(2)

INPUT #1, leader$
INPUT #1, Euler(3)
PRINT leader$; Euler(3)

INPUT #1, leader$
INPUT #1, NoseForwTemp
PRINT leader$; NoseForwTemp

INPUT #1, leader$
INPUT #1, NoseRearTemp
PRINT leader$; NoseRearTemp

INPUT #1, leader$
INPUT #1, NoseEmis
PRINT leader$; NoseEmis

INPUT #1, leader$
INPUT #1, BodyForwTemp
PRINT leader$; BodyForwTemp

INPUT #1, leader$
INPUT #1, BodyRearTemp
PRINT leader$; BodyRearTemp

INPUT #1, leader$
INPUT #1, BodyEmis
PRINT leader$; BodyEmis

INPUT #1, leader$
INPUT #1, FinOuterWRTInnerTemp
PRINT leader$; FinOuterWRTInnerTemp

INPUT #1, leader$
INPUT #1, FinLeadTemp
```

```
    PRINT leader$; FinLeadTemp

    INPUT #1, leader$
    INPUT #1, FinTrailTemp
    PRINT leader$; FinTrialTemp

    INPUT #1, leader$
    INPUT #1, FinEmis
    PRINT leader$; FinEmis

    INPUT #1, leader$
    INPUT #1, AftBodyForwTemp
    PRINT leader$; AftBodyForwTemp

    INPUT #1, leader$
    INPUT #1, AftBodyRearTemp
    PRINT leader$; AftBodyRearTemp

    INPUT #1, leader$
    INPUT #1, AftBodyEmis
    PRINT leader$; AftBodyEmis

    INPUT #1, leader$
    INPUT #1, File1$
    PRINT leader$; File1$

    INPUT #1, leader$
    INPUT #1, Scale
    PRINT leader$; Scale

    INPUT #1, leader$
    INPUT #1, WindowX1
    PRINT leader$; WindowX1

    INPUT #1, leader$
    INPUT #1, WindowY1
    PRINT leader$; WindowY1

    INPUT #1, leader$
    INPUT #1, WindowX2
    PRINT leader$; WindowX2

    INPUT #1, leader$
    INPUT #1, WindowY2
    PRINT leader$; WindowY2
    PRINT
                                          'number of lines of data that describes
                                          'each facet is determined
    NumFacets% = 0
    DO
       INPUT #1, Test$
       NumFacets% = NumFacets% + 1
       LOOP WHILE Test$ <> "END"
    NumFacets% = (NumFacets% - 1) / 2
                                          'datafile is closed
    CLOSE #1

END SUB

SUB ReadInputData2
'*******************************************************************************************************************

'THIS SUBROUTINE MAKES THE SECOND PASS THROUGH THE PREVIOUSLY GENERATED
DATA
'FILE THAT DESCRIBES THE PENETRATOR GEOMETRY AND THERMAL PROPERTIES. DURING
```

```
'THIS PASS THE LEADER INFORMATION IS IGNORED BUT THE SUBSEQUENT LINES THAT
'DEFINE THE INDIVIDUAL FACET PARAMETERS ARE INPUT
'*******************************************************************************************************

   SHARED File1$, NumFacets%, Type$(), X(), Y(), Z(), NormalCos()
   SHARED Area(), Temp(), Emis()

   OPEN File1$ FOR INPUT AS #1                          'datafile is reopened

   INPUT #1, leader$                                    'leader data is input but not used
   INPUT #1, LengthNoseCone

   INPUT #1, leader$
   INPUT #1, DeltaZNoseCone

   INPUT #1, leader$
   INPUT #1, NumNoseConeLongSeg%

   INPUT #1, leader$
   INPUT #1, RadNoseCone

   INPUT #1, leader$
   INPUT #1, NumNoseConeRadSeg%

   INPUT #1, leader$
   INPUT #1, LengthBody

   INPUT #1, leader$
   INPUT #1, DeltaZBody

   INPUT #1, leader$
   INPUT #1, NumBodyLongSeg%

   INPUT #1, leader$
   INPUT #1, RadBody

   INPUT #1, leader$
   INPUT #1, NumBodyRadSeg%

   INPUT #1, leader$
   INPUT #1, NumFins%

   INPUT #1, leader$
   INPUT #1, ThickFin

   INPUT #1, leader$
   INPUT #1, LengthBaseFin

   INPUT #1, leader$
   INPUT #1, LengthLeadEdgeFin

   INPUT #1, leader$
   INPUT #1, DeltaZFin

   INPUT #1, leader$
   INPUT #1, NumLeadEdgeLongSeg%

   INPUT #1, leader$
   INPUT #1, NumNonLeadEdgeLongSeg%

   INPUT #1, leader$
   INPUT #1, HeightFin

   INPUT #1, leader$
   INPUT #1, DeltaRadFin
```

```
INPUT #1, leader$
INPUT #1, NumFinRadSeg%

INPUT #1, leader$
INPUT #1, NumAftBodyRadSegPerFin%

INPUT #1, leader$
INPUT #1, Euler

INPUT #1, leader$
INPUT #1, Euler

INPUT #1, leader$
INPUT #1, Euler

INPUT #1, leader$
INPUT #1, NoseForwTemp

INPUT #1, leader$
INPUT #1, NoseRearTemp

INPUT #1, leader$
INPUT #1, NoseEmis

INPUT #1, leader$
INPUT #1, BodyForwTemp

INPUT #1, leader$
INPUT #1, BodyRearTemp

INPUT #1, leader$
INPUT #1, BodyEmis

INPUT #1, leader$
INPUT #1, FinOuterWRTInnerTemp

INPUT #1, leader$
INPUT #1, FinLeadTemp

INPUT #1, leader$
INPUT #1, FinTrailTemp

INPUT #1, leader$
INPUT #1, FinEmis

INPUT #1, leader$
INPUT #1, AftBodyForwTemp

INPUT #1, leader$
INPUT #1, AftBodyRearTemp

INPUT #1, leader$
INPUT #1, AftBodyEmis

INPUT #1, leader$
INPUT #1, File1$

INPUT #1, leader$
INPUT #1, Scale

INPUT #1, leader$
INPUT #1, WindowX1

INPUT #1, leader$
INPUT #1, WindowY1
```

```
    INPUT #1, leader$
    INPUT #1, WindowX2

    INPUT #1, leader$
    INPUT #1, WindowY2
                                                    'data about the geometry and thermal
                                                    'properties of each facet is input
    FOR I% = 1 TO NumFacets%
       INPUT #1, Type$(I%), X(I%, 1), Y(I%, 1), Z(I%, 1), X(I%, 2), Y(I%, 2), Z(I%, 2), X(I%, 3),
                                      Y(I%, 3), Z(I%, 3), X(I%, 4), Y(I%, 4), Z(I%, 4),
                                      NormalCos(I%), Area(I%), Temp(I%), Emis(I%)
       NEXT I%
                                                    'datafile is closed
    CLOSE #1

END SUB
```

INTENTIONALLY LEFT BLANK.

NO. OF COPIES	ORGANIZATION
2	DEFENSE TECHNICAL INFO CTR ATTN DTIC DDA 8725 JOHN J KINGMAN RD STE 0944 FT BELVOIR VA 22060-6218
1	DIRECTOR US ARMY RESEARCH LAB ATTN AMSRL OP SD TA 2800 POWDER MILL RD ADELPHI MD 20783-1145
3	DIRECTOR US ARMY RESEARCH LAB ATTN AMSRL OP SD TL 2800 POWDER MILL RD ADELPHI MD 20783-1145
1	DIRECTOR US ARMY RESEARCH LAB ATTN AMSRL OP SD TP 2800 POWDER MILL RD ADELPHI MD 20783-1145
	ABERDEEN PROVING GROUND
2	DIR USARL ATTN AMSRL OP AP L (305)

NO. OF COPIES	ORGANIZATION
5	DIRECTOR US ARMY RESEARCH LAB ATTN AMSRL PS V GELNOVATCH R HAMLEN J KEY AMSRL PS D M TOMPSETT AMSRL PS T R REITMEYER BLDG 2700 FT MONMOUTH NJ 07703-5601
1	DIRECTOR US ARMY RESEARCH LAB ATTN AMSRL SL E G MARES WSMR NM 88002-5501
2	VHCLE STRCTRS DIRECTORATE ATTN AMSRL VS L D HOD AMSRL VS S F BARTLETT MS 266 NASA LANGLEY RSRCH CTR HAMPTON VA 23681-0001
2	VHCLE PRPLSN DIRECTORATE ATTN AMSRL VP C R BILL AMSRL VP T G BOBULA 21000 BROOKPARK RD CLEVELAND OH 44135-3191

NO. OF COPIES	ORGANIZATION
	ABERDEEN PROVING GROUND
25	DIR, USARL ATTN: AMSRL-MA, J. DIGNAM (CNR) AMSRL-MA-A, D. VIECHNICKI (CNR) AMSRL-SE, J. MILLER (ALC) AMSRL-SE-E, J. PELLEGRINO (ALC) AMSRL-WT-N, J. INGRAM (ALC) AMSRL-SL-B, P. DEITZ AMSRL-SL-C, W. HUGHES AMSRL-WT-P, A. HORST AMSRL-WT-T, W. MORRISON AMSRL-WT-W, C. MURPHY, JR. AMSRL-WT-WD, A. NIILER P. BERNING R. BOSSOLI S. CORNELISON A. GAUSS, JR. C. HOLLANDSWORTH C. HUMMER L. KECSKES T. KOTTKE M. MCNEIR J. POWELL A. PRAKASH D. STRENZWILK C. STUMPFEL G. THOMSON

USER EVALUATION SHEET/CHANGE OF ADDRESS

This Laboratory undertakes a continuing effort to improve the quality of the reports it publishes. Your comments/answers to the items/questions below will aid us in our efforts.

1. ARL Report Number/Author ARL-MR-329 (Kottke) ______ Date of Report August 1996

2. Date Report Received ______

3. Does this report satisfy a need? (Comment on purpose, related project, or other area of interest for which the report will be used.) ______

4. Specifically, how is the report being used? (Information source, design data, procedure, source of ideas, etc.) ______

5. Has the information in this report led to any quantitative savings as far as man-hours or dollars saved, operating costs avoided, or efficiencies achieved, etc? If so, please elaborate. ______

6. General Comments. What do you think should be changed to improve future reports? (Indicate changes to organization, technical content, format, etc.) ______

CURRENT ADDRESS

Organization

Name

Street or P.O. Box No.

City, State, Zip Code

7. If indicating a Change of Address or Address Correction, please provide the Current or Correct address above and the Old or Incorrect address below.

OLD ADDRESS

Organization

Name

Street or P.O. Box No.

City, State, Zip Code

(Remove this sheet, fold as indicated, tape closed, and mail.)
(DO NOT STAPLE)

DEPARTMENT OF THE ARMY

OFFICIAL BUSINESS

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO 0001,APG,MD

POSTAGE WILL BE PAID BY ADDRESSEE

DIRECTOR
US ARMY RESEARCH LABORATORY
ATTN AMSRL WT WD
ABERDEEN PROVING GROUND MD 21005-5066